

VALORACIÓN DE INMUEBLES MEDIANTE TÉCNICAS DE LÓGICA DIFUSA



UNIVERSIDAD COMPLUTENSE DE MADRID

PROYECTO DE SISTEMAS INFORMÁTICOS
CURSO ACADÉMICO: 2007-2008
FACULTAD DE INFORMÁTICA

CARLOS FERNÁNDEZ AGÜERO
IGNACIO FERNÁNDEZ CUESTA
DAVID MOYA COLLADOS

DIRIGIDO POR: DRA. DÑA. **MARÍA VICTORIA LÓPEZ LÓPEZ**
DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

Nosotros, los autores del proyecto Valoración de Inmuebles mediante Técnicas de Lógica Difusa, de la asignatura de *Sistemas Informáticos*:

Carlos Fernández Agüero con DNI: 02.280.501 - M

Ignacio Fernández Cuesta con DNI: 75.110.760 - M

David Moya Collados con DNI: 02.666.706 - V

Dirigidos por:

Dra. María Victoria López López

Departamento de Arquitectura de Computadores y Automática

Autorizamos a la Universidad Complutense de Madrid a utilizar y difundir, con fines académicos, el contenido de este documento de texto, así como del contenido del CD complementario que adjuntamos con él mismo.

Carlos Fernández Agüero

Ignacio Fernández Cuesta

David Moya Collados

Índice

1. Introducción al Problema
2. Objetivo del Proyecto
3. Estado del Arte
 - 3.1. Sector Inmobiliario
 - 3.1.1. Introducción contextual
 - 3.1.2. Estado del mercado
 - 3.2 La lógica fuzzy
 - 3.2.1. Introducción
 - 3.2.2. Herramientas
 - 3.2.3. XFuzzy
4. Sistema de Valoración de Inmuebles mediante técnicas de lógica difusa
 - 4.1. Descripción Funcional
 - 4.2. Desarrollo del Sistema
 - 4.2.1. Definición del sistema
 - 4.2.2. Desarrollo del Motor de Inferencia
 - 4.2.3. Desarrollo de la BBDD
 - 4.2.4. Desarrollo de la Aplicación
 - 4.3. Pruebas
 - 4.3.1. Ejecución de las pruebas
 - 4.3.2. Resultados
 - 4.4. Trabajo futuro
5. Conclusiones
6. Manual del Usuario

7. Apéndices

7.1. Instalación de XFuzzy 3.0

7.2. Instalación de Eclipse

8. Palabras clave

9. Bibliografía

1. Introducción al Problema

El presente proyecto tiene como objetivo presentar una aplicación práctica real del concepto de valoración comercial llevada a cabo mediante técnicas de lógica difusa.

En la actualidad, las tendencias en el campo de la gestión comercial apuntan a la necesidad de complementar el proceso tradicional de tasación, mayoritariamente económica, con otras ideas o conceptos alternativos si deseamos alcanzar el éxito. Es aquí donde se propone el empleo de una variable adicional de análisis, *la valoración*, basada en un sistema de clasificación sobre el sector comercial al que estemos dirigidos. Para su obtención se emplea la lógica difusa, que nos ofrece una representación estructurada del conocimiento y una facilidad de manipulación de información de forma numérica que hace que constituya de esta forma la herramienta ideal para la construcción de nuestro sistema de valoración.

Siguiendo el camino marcado por trabajos incluidos en proyectos como 'Sistemas de clasificación borrosa para la toma de decisiones basados en modelos operacionales de agregación' aprobada por el Ministerio de Ciencia y Tecnología con código: DGI, MTM 2005-08982-C04-01, dirigido por el profesor doctor Javier Montero, como continuación a la evaluación de sentencias con propiedades borrosas [FLINS06] y aplicación paralela en [CEDI07, FLINS08], se encuentra nuestro proyecto, una aplicación mediante la cual obtener el sistema con la finalidad anteriormente descrita y aplicarlo a una base de datos reales y actuales con una fuerte componente difusa, sin renunciar a las aportaciones de los datos nítidos y al análisis estadístico tradicional.

El sector comercial que se nos propone para nuestro proyecto, es el sector inmobiliario. Teniendo en cuenta todo esto, nuestra aplicación actuará de la siguiente manera: dado un inmueble y un tipo de comprador, de entre una serie de perfiles representativos obtenidos de una clasificación del sector poblacional facilitados por un

experto, queremos obtener la valoración comercial que dicho inmueble tiene para ese comprador.

Nótese en este punto, la ventaja de nuestro sistema respecto a una tasación tradicional, ya que está comprobado que el potencial cliente/comprador, a la hora de elegir un inmueble u otro, desea que lo que busca cumpla una serie de requisitos que tiene entre sus preferencias. Por ello, para encontrar el inmueble que mejor se adapta a cada tipo de comprador, tendemos que comprobar en que medida cumple ese inmueble con los requisitos dados por el comprador.

El siguiente bloque importante que conforma nuestro proyecto es pues, el empleo de la lógica difusa. Para realizar el cálculo matemático de la valoración comercial que tiene cada inmueble para cada uno de los distintos tipos de compradores descritos con anterioridad, es requisito utilizar algún tipo de sistema formal, ante la necesidad de modelar dicho cálculo y facilitar la tarea de especificación de requerimientos inherente a cualquier proyecto de ingeniería. Y aquí es donde entra en juego la lógica difusa.

Para realizar las distintas tareas de modelado, representación y cálculos relacionados con la lógica fuzzy, hay disponibles una serie de herramientas encaminadas a facilitar y automatizar en la medida de lo posible el proceso de diseño del sistema de lógica difusa. Tras estudiar las opciones a nuestro alcance, se tomó la decisión de utilizar una herramienta denominada XFuzzy, desarrollada por el Instituto de Microelectrónica de Sevilla.

Esta herramienta, además de ofrecer un entorno de diseño y desarrollo de sistemas borrosos, también permite generar automáticamente el esqueleto de una implementación en diferentes lenguajes de programación como Java, C o C++. Sobre este aspecto, en la implementación de la aplicación, nuestra elección para su desarrollo ha sido la plataforma Java.

2. Objetivo del Proyecto

RESUMEN

El proyecto propuesto consiste en la aplicación práctica de la lógica difusa en el campo del sector inmobiliario. El objetivo de la misma es obtener una valoración comercial de cada inmueble, clasificando por orden de idoneidad en perfiles de potenciales compradores una serie de inmuebles almacenados en una base de datos categorizados en función de una serie de parámetros inmobiliarios, como podrían ser la tasación económica del inmueble, el tipo de inmueble, su localización, la orientación o la luminosidad del mismo.

La aplicación dispone de un interfaz gráfico para interactuar con el usuario, y tras acceder a la información almacenada en la base de datos a la cual está conectada, mediante un motor de inferencia de aplicación de lógica difusa, se obtendrá un cálculo aproximado de la valoración que cada inmueble tiene para un perfil determinado, mostrando los resultados al usuario.

ABSTRACT

The current Project consists of a practical application of fuzzy logic in the field of the real estates sector. The objective involves obtaining a commercial appraisal of each estate stored in a database, classifying them while taking into account its suitability to different potential customer profiles. The system categorizes the estates in terms of specific technical factors such as economic taxation, type of estate, its location, accessibility, orientation or luminosity.

The application provides a graphic interface appointed to interact with the user, and once accessed the information stored in the database connected to the system, a fuzzy inference engine is used to show an approximated calculation of the valuation of the estates for the selected profile.

3. Estado del Arte

3.1. Sector Inmobiliario

3.1.1. Introducción contextual

Como se ha expuesto anteriormente, nuestro proyecto pretende realizar una aplicación de la lógica difusa a una cierta problemática englobada en el sector inmobiliario. Con este apartado se pretende aportar una visión general del estado del sector en la sociedad española, con objetivo de comprender el alcance e idoneidad de las funcionalidades aportadas.

Para hacernos a la idea del potencial de explotación en forma de desarrollos como el que nos ocupa, orientados a complementar los sistemas de gestión comercial existentes, es necesario conocer la problemática. Hasta el momento, el sector inmobiliario ha estado dominado por los sistemas de tasación puramente económicos, que si bien tienen éxito a la hora de estudiar y estimar el valor monetario y material de una propiedad o vivienda concreta y están muy extendidos en forma de organizaciones o sociedades especializadas, aplicaciones telemáticas o incluso procedimientos manuales como formularios, desde un punto de vista comercial, en la actualidad, no resultan suficientes.

No obstante como se ha expresado anteriormente, la utilidad y experiencia en el mercado de estos sistemas de tasación existentes queda fuera de toda duda, y por tanto, no sólo no enfrentamos la funcionalidad que presenta nuestro proyecto con estos sistemas, sino que al mismo tiempo que buscamos complementarlos, hacemos nuestros algunos de los procedimientos y fundamentos estadísticos en los que se basan dichos sistemas.

3.1.2. Estado del mercado

A continuación se pretende presentar de forma escueta algunas de las tendencias existentes en auge aplicadas en gestión comercial inmobiliaria, de esta forma se facilita un marco de conocimiento de algunas alternativas con funcionalidad similar a nuestro proyecto (con la entendible diferencia de alcance), pudiendo comparar y valorar en mejor medida las desventajas y puntos fuertes de la solución presentada, así como crear el trasfondo necesario para poder plantearse posibles mejoras o complementos futuros.

Comenzaremos por una de las herramientas que mayor importancia y proyección posee en la actualidad: los buscadores de producto.

Buscadores de Producto

Los buscadores de producto son sistemas de información que ayudan al consumidor en la tarea de identificar productos de entre una gran oferta de alternativas similares.

El grado de complejidad de un buscador de producto va desde los más sencillos algoritmos de búsqueda hasta algunos de los más complejos denominados sistemas de soporte a decisiones (DSS), cuya filosofía es uno de los pilares de nuestro proyecto. Un DSS puede adoptar muchas formas diferentes, pero en general, podemos decir que se trata de un sistema utilizado para servir de apoyo, más que automatizar, el proceso de toma de decisiones. La decisión es una elección entre alternativas basadas en estimaciones de los valores de esas alternativas. Como puede comprobarse, nuestra aplicación se ajusta bastante a este esquema, gestionando la toma de decisión mediante un sistema lógico-difuso.

Normalmente los buscadores de productos forman parte de iniciativas comerciales, ya sea de forma directa o indirecta. Conceptualmente un buscador de producto tiene como objetivo desembocar en una compra o en proporcionar un interés de compra.

Los buscadores de producto están fundamentalmente diseñados para operar sobre grupos de producto en los que se comparten características y criterios específicos, como es el caso de productos tecnológicos (televisores, ordenadores portátiles...) o una base de datos de inmuebles en nuestro proyecto.

Como ejemplo de aplicación con éxito de muchos de los conceptos anteriormente citados, tenemos entre otras a www.idealista.com, compañía creada en octubre de 2000 que ofrece a través de Internet servicios sobre contenidos inmobiliarios a nivel nacional. El portal se ha consolidado como uno de los sitios web líderes no sólo de compraventa y arrendamiento de inmuebles.

Como hemos podido concluir de la argumentación aportada en las anteriores secciones, es necesaria en gran medida la disponibilidad de una base fuerte y estable de información sobre la que trabajar de un modo comercial, y en el caso de Idealista.com, el Bilbao Bizkaia Kutxa tiene el papel de accionista de referencia y ofrece el acceso a una base de datos del mercado inmobiliario, que integra la oferta de particulares y profesionales inmobiliarios. Cada anuncio de idealista cuenta, al menos, con 40 características de la vivienda, creando así un conjunto sólido de características de referencia sobre las que articular una selección aceptable en función de diferentes criterios. Aquí es donde entran en juego conceptos derivados de sistemas de recomendación que se sirven de la gran red de usuarios del sistema.

No obstante, la aplicación de los buscadores de producto no se limita a productos físicos, sino también al mundo de los servicios, como por ejemplo tipos de cuentas bancarias, proveedores de comunicación, etc.

Existe una corriente importante que busca la integración de los buscadores de producto en el mundo emergente de las redes sociales, con el objetivo de proporcionar un dinamismo en el añadido de productos y valoraciones en lo que podríamos denominar un acercamiento a la filosofía de los muy pujantes sistemas de recomendación.

Sistemas de Recomendación

Los sistemas de recomendación conforman un tipo de técnica especial de filtro de información que intenta presentar la información (ya sea en forma de información pura, o en forma de productos desde un punto de vista comercial) al usuario en función de lo que podría llegar a interesarle. Normalmente, un sistema de recomendación compara el perfil de usuario (que se conoce o es inferido) con una serie de características de referencia. Estas características pueden hacer referencia al producto o información (aproximación basada en el contenido), o al marco de información en el que se engloba el perfil del usuario (aproximación basada en filtro colaboracional).

A la hora de construir el perfil del usuario como los usados en nuestra aplicación, hay varias formas de recogida de datos que se agrupan principalmente en implícitas y explícitas. Actualmente cobran importancia los métodos de recogida implícitos, conformados por seguimientos y registros dinámicos del comportamiento del potencial cliente. La mayoría de la gente es más predecible de lo que ella cree. Cuando el ser humano se encuentra en una determinada situación, es altamente probable que reaccione de una manera predeterminada. Si observamos a los clientes por cierto tiempo, es fácil descubrir fuertes regularidades en su comportamiento, por ejemplo, en como realiza la búsqueda de información. En este contexto las tecnologías de la información tienen mucho que decir.

Un ejemplo muy importante en la actualidad, es el dominio Amazon. Amazon, emplea mucho tiempo en investigación relacionada con sistemas de recomendación. Allí, estos sistemas se basan en lo que el usuario hace mientras navega y en lo que compra tanto ese cliente como otros, todos ellos reales. En base a eso se utiliza la información para recomendarle otros productos que puedan gustarle: el clásico «si te gustó esto, entonces te gustará...». La ventaja de una tienda como Amazon es que cuenta con toda la información de miles y miles de compras cada día, cuyos datos puede recolectar y analizar, y en Amazon se *confía* en ese análisis.

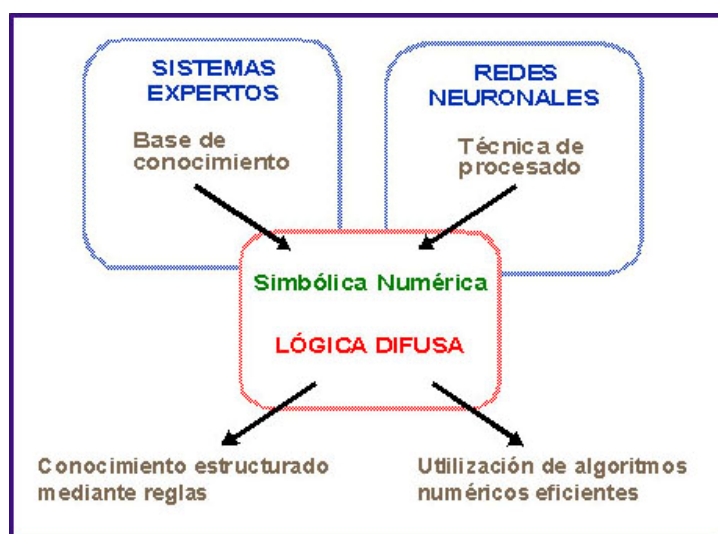
3.2. La lógica fuzzy

3.2.1. Introducción

La presente sección pretende abarcar un resumen de los conceptos clave de la lógica borrosa proporcionando una idea de la potencia y posibilidades de aplicación del paradigma a problemáticas de este tipo, intentando ceñirnos al marco y alcance de nuestro proyecto.

La lógica borrosa emerge como una herramienta interesante para el control y representación de subsistemas y procesos industriales complejos, así como también para la electrónica de entretenimiento y hogar, sistemas de diagnóstico y otros sistemas expertos. Con este sistema formal se pretende representar de forma rigurosa el significado de los enunciados imprecisos del lenguaje natural.

Al igual que los sistemas expertos y las redes neuronales, los sistemas basados en lógica difusa obtienen la salida del sistema en función de sus entradas sin necesidad de recurrir a la utilización de un modelo analítico. Comparten con los primeros la característica de representar el conocimiento de forma estructurada (mediante reglas) y con los segundos la facilidad para manipular la información de forma numérica. Esta última cualidad hace especialmente atractiva la implementación en hardware de sistemas de inferencia basados en lógica difusa.



La lógica difusa se utiliza cuando la complejidad del proceso en cuestión es muy alta y no existen modelos matemáticos precisos, para procesos altamente no lineales y cuando están involucradas definiciones y conocimiento no estrictamente definido (impreciso o subjetivo) asociado al lenguaje natural.

En lógica clásica una proposición sólo admite dos valores: verdadero o falso. Por ello se dice que es una lógica bivalente o binaria. La lógica difusa (o borrosa) es otro tipo de lógica, que se caracteriza por querer cuantificar esta incertidumbre: Si P es una proposición, se le puede asociar un número $v(P)$ en el intervalo $[0,1]$ tal que:

1. Si $v(P) = 0$, P es falso.
2. Si $v(P) = 1$, P es verdadero.
3. La veracidad de P aumenta con $v(P)$.

El aspecto central de este sistema formal, de modo distinto a la lógica clásica de sistemas, radica en la modelización de modos de razonamiento imprecisos, los cuales juegan un rol esencial en la destacable habilidad humana de trazar decisiones racionales en un ambiente de incertidumbre e imprecisión. Esta habilidad depende, en cambio, de nuestra habilidad de inferir una respuesta aproximada a preguntas basadas en un conjunto de conocimiento que es inexacto, incompleto o no totalmente confiable. Por lo tanto, para llevar a cabo con éxito esta tarea, es necesario poseer una especialización y conocimiento técnico en el contexto en el que nos movamos. En nuestro proyecto esta labor viene proporcionada por un experto en el sector inmobiliario, cuya labor explicitaremos más adelante.

Sumergiéndonos un poco en la historia, tenemos que esta idea nació en un artículo de Lotfi A. Zadeh publicado en 1965 y titulado "Fuzzy Sets" (Conjuntos Difusos) ["Fuzzy Sets". Zadeh, 1965]. Pero hay que tener en cuenta que la idea de la parametrización "borrosa" de las cosas, se hereda desde la época de los primeros grandes filósofos. Posteriormente a ellos, otros grandes pensadores como David Hume o Kant apoyaban esta idea manteniendo que el razonamiento venía dado por las

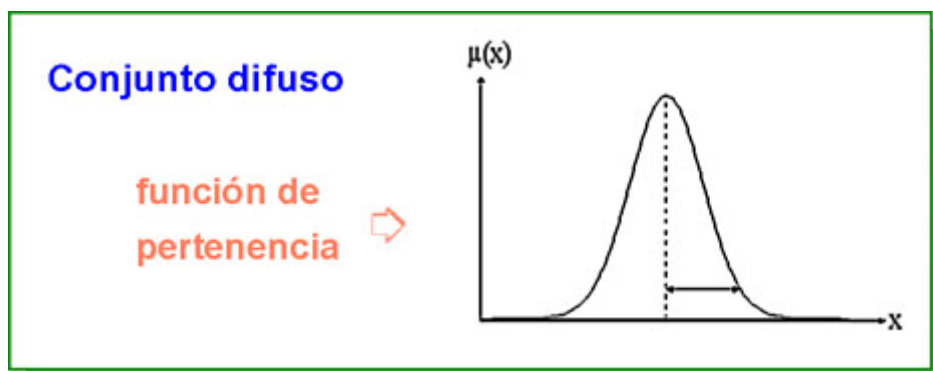
observaciones de las que somos testigos a lo largo de nuestra vida y la detección de algunos principios contradictorios en la lógica clásica.

A continuación repasamos algunos de los pilares de la lógica difusa con los que hemos trabajado directamente durante el desarrollo del proyecto:

Conjuntos difusos:

El concepto fundamental en que se basa la lógica difusa es el de conjunto difuso, un tipo de conjunto caracterizado porque los elementos del universo de discurso en el que está definido pueden pertenecer a él en un cierto grado, representado por una función de pertenencia.

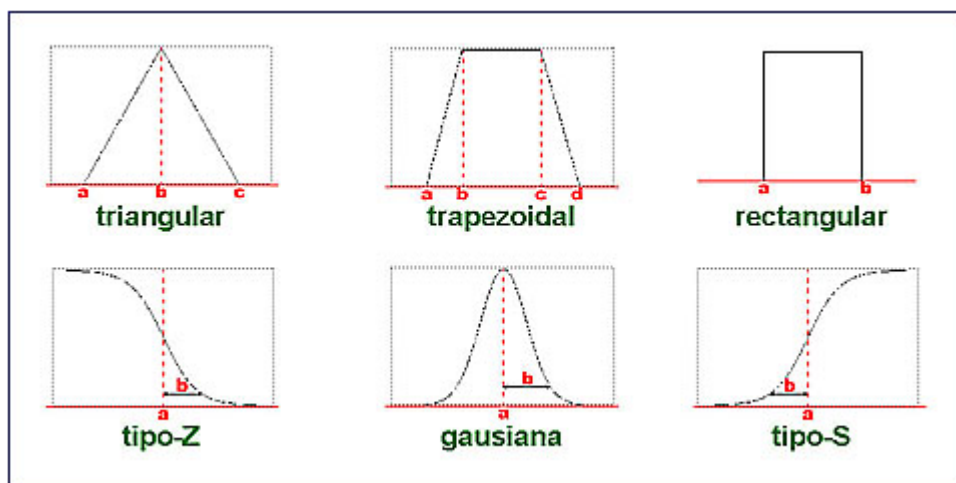
Zadeh en 1965 definió el concepto de conjunto difuso basándose en la idea del grado de pertenencia. Por ejemplo, el conjunto de las personas que son altas es un conjunto difuso, pues no está claro el límite de altura que establece a partir de que medida una persona es alta o no lo es. Ese límite es difuso y, por tanto, el conjunto que delimita también lo será.



Un conjunto difuso A sobre un universo de discurso U es un conjunto de pares dado por:

$$A = \{ \mu_A(u) / u : u \in U, \mu_A(u) \in [0,1] \}$$

Donde, μ es la llamada función de pertenencia y $\mu_A(u)$ es el grado de pertenencia del elemento u al conjunto difuso A . La función de pertenencia $\mu_A(u)$ describe, por tanto, el grado de pertenencia de los diferentes elementos del universo de discurso al conjunto difuso. Este grado oscila entre los extremos 0 y 1, $\mu_A(u) = 0$, indica que u no pertenece en absoluto al conjunto difuso A , $\mu_A(u) = 1$, indica que u pertenece totalmente al conjunto difuso A . La elección de la forma de la función de pertenencia es subjetiva y dependiente del contexto. No obstante, por razones prácticas, en la literatura se suelen emplear funciones triangulares, trapezoidales o en forma de campana como las que se muestran en la figura.



Relaciones difusas:

El concepto de relación difusa es una generalización del concepto de relación de la teoría clásica de conjuntos. Mientras que una relación entre dos conjuntos clásicos describe la existencia o no de asociación entre los elementos de ambos conjuntos, una relación difusa describe el grado de asociación o interacción entre los elementos de dos o más conjuntos difusos.

En el caso discreto, la relación difusa puede representarse mediante una matriz, denominada matriz relacional difusa, cuyos elementos toman valores que se encuentran en el intervalo $[0,1]$.

Atributos difusos:

Los atributos difusos se clasifican en tres tipos:

Tipo 1: estos atributos son “datos precisos” (clásicos, sin imprecisión) que pueden tener etiquetas lingüísticas definidas sobre ellos. Los atributos de Tipo 1 reciben una representación igual que los datos precisos, pero puedan ser manejados en condiciones difusas. Por ejemplo, la estatura de una persona.

Tipo 2: son atributos que pueden almacenar o tomar datos imprecisos sobre un conjunto de referencia ordenado. Estos atributos admiten tanto datos clásicos como difusos, en forma de distribuciones de posibilidad. Por ejemplo, la edad puede tener las etiquetas niño, joven, adulto, referenciadas sobre un conjunto entre 0 y 100.

Tipo 3: son atributos sobre datos imprecisos sobre un conjunto de referencia normalizado pero no ordenado. Estos atributos son definidos sobre un dominio subyacente no ordenado, por ejemplo, el atributo “color del pelo” puede tener las etiquetas rubio, pelirrojo y castaño.

Nuestra propuesta engloba el uso de todos los tipos de atributos difusos, Tipo 1, 2 y 3 para caracterizar los conceptos del esquema del dominio, ya que los conceptos pueden ser tratados como clases de un modelo orientado objeto.

Cuantificadores difusos:

Los cuantificadores difusos permiten expresar cantidades o proporciones difusas para dar una idea aproximada del número de elementos de un subconjunto (o que cumplen cierta condición) o de la proporción de ese número en relación con el total de elementos posibles. Los cuantificadores pueden ser absolutos o relativos:

Cuantificadores absolutos: expresan cantidades sobre el número total de elementos de un determinado conjunto, diciendo si este número es “grande”, “muchísimos”, “aproximadamente entre 5 y 10”, etc.

Cuantificadores relativos: expresan mediciones sobre el número total de elementos que cumplen cierta característica dependiendo del total de elementos posibles, por lo que la verdad del cuantificador depende de dos cantidades. Este tipo de cuantificadores se usa en expresiones como “la mayoría”, “la minoría”, “aproximadamente 40 años”.

Base de reglas:

Un sistema de inferencia basado en lógica difusa, o sistema difuso, está formado por un conjunto de reglas del tipo “si x es A entonces y es B”, donde x e y son las variables del sistema, y A y B son términos lingüísticos como ‘alto’, ‘bajo’, ‘medio’, ‘positivo’, ‘negativo’, etc.

Reglas si-entonces

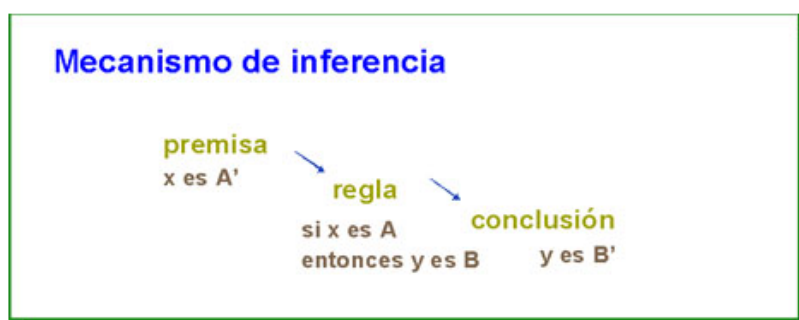
*"SI la velocidad es alta Y
el obstáculo está cerca ENTONCES
la fuerza de frenado debe ser grande "*

Mecanismo de inferencia:

Las técnicas de razonamiento utilizadas en lógica difusa son una generalización de los mecanismos de inferencia empleados en la lógica bi-valuada tradicional, pero a diferencia de un sistema experto convencional, en un sistema difuso varias reglas pueden estar activas simultáneamente con diferentes grados de activación. Cuando varía una de sus entradas el sistema evalúa todas las reglas para inferir una conclusión o salida. La regla composicional de inferencia proporciona un mecanismo para evaluar el resultado de una regla difusa. Un sistema de inferencia difuso contendrá un conjunto de

reglas de descripción lingüística. En el caso más general los antecedentes y consecuentes de estas reglas incluirán proposiciones difusas compuestas, es decir, combinarán múltiples entradas y salidas. Este tipo de sistema se denomina sistema MIMO (“multiple input multiple output”).

No obstante, un sistema MIMO siempre puede ser considerado como un conjunto de sistemas con entradas múltiples y una única salida, o sistemas MISO (“multiple input single output”).

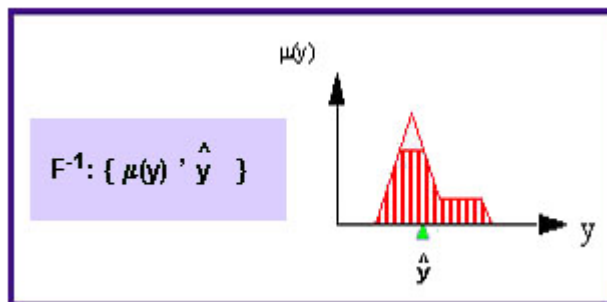


Dentro de estas técnicas, se encuentran, por ejemplo, el mecanismo de inferencia Min-Max (o método de Mamdani) o el método Producto-Suma.

Métodos de defuzzyficación:

En los mecanismos de inferencia anteriormente citados, el resultado de la inferencia es un conjunto difuso. Para que esta información pueda utilizarse en determinadas aplicaciones, como las de control, es preciso obtener un valor concreto representativo de dicho conjunto.

El proceso de defuzzyficación se expresa mediante el operador defuzzificador F^{-1} que transforma la función de pertenencia representativa de un conjunto difuso $\mu(y)$ en un elemento concreto del universo de discurso:



Algunos de los métodos de defuzzificación existentes y empleados durante el proyecto son el Centro de Gravedad (CoG), el Centro de Sumas (CoS), el Primer Máximo (FoM), el Último Máximo (LoM), la Media de Máximos (MoM), la Media Difusa (FM), la Media Difusa Ponderada (WFM), el Método de Calidad (QM), el Método “Level Grading” (LGM) o el Método de Yager (YM).

Para la elaboración de esta sección se han consultado numerosas fuentes de información de las cuales hacer aquí un registro sería demasiado extenso, sobre todo en forma de dominios web, por lo que nos limitamos a resaltar la importancia que ha tenido la publicación electrónica del Instituto de Microelectrónica de Sevilla FLEB o Fuzzy Logic E-Book [FLEB. IMSE-CNM, 2001].

3.2.2. Herramientas

La investigación en torno a los sistemas difusos es tan activa que se producen numerosos avances prácticos y teóricos en este campo. Los primeros sistemas difusos fueron controladores con un par de reglas con simples cláusulas “if-then” obtenidas del conocimiento heurístico. Actualmente, los sistemas basados en lógica difusa pueden contener una gran diversidad de módulos de decisión, clasificadores difusos, controladores difusos... que pueden ser combinados con otros módulos o tecnologías no difusas con el objetivo de intercambiarse información enriqueciendo el proceso. En dicho proceso, las reglas empleadas pueden ser reforzadas mediante diferentes valores y

pueden usar diferentes funciones de pertenencia, así como operadores, para relacionar los antecedentes y los consecuentes y obtener la conclusión global. Además, las reglas no son sólo obtenidas del conocimiento heurístico, sino también del procesamiento numérico de información (campo atrayente en la actualidad), y de la aplicación de simplificación. En general, el refinamiento de métodos para obtener reglas, así como procesos de mejora de éstas, figuran entre las prácticas habituales hoy en día.

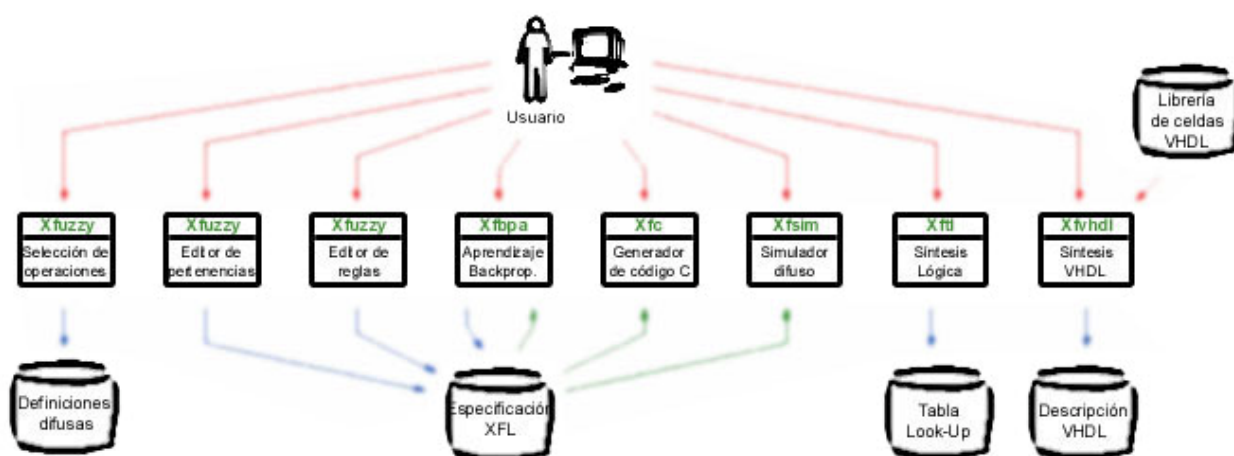
A la hora de decidirnos por una herramienta determinada de las que se ofrecen en la actualidad, un factor de decisión clave fue la experiencia de realización previa de desarrollos bajo la herramienta XFuzzy en propuestas anteriores pertenecientes a la iniciativa en la que se engloba nuestro proyecto. No obstante se evaluaron diferentes alternativas para la construcción del motor de inferencia de la aplicación, pudiendo observar en términos generales como se explicaba al comienzo de este apartado, que el gran desarrollo y evolución del campo no permite la existencia de la herramienta de lógica difusa definitiva.

Un proyecto que merece la pena resaltar aparte del ya mencionado XFuzzy, por su filosofía y pujanza es jFuzzyLogic, un paquete en desarrollo, de código libre, que aporta directamente al entorno Java en el que nos desenvolvamos, una gran serie de funcionalidades propias de la lógica difusa que tienen potencial para la construcción de sistemas ciertamente complejos.

3.2.3. *XFuzzy*

A continuación ofrecemos una descripción de la herramienta difusa empleada como pilar fundamental en la implementación del motor de inferencia de la aplicación que conforma nuestro proyecto. En un principio, la herramienta XFuzzy estaba orientada a la descripción y simulación de simples controladores difusos. El paso del tiempo y el incremento en complejidad que tratábamos en el apartado anterior motivaron una evolución en la oferta de funcionalidades del entorno XFuzzy.

La versión actual, XFuzzy 3, contiene un conjunto de herramientas CAD que comparten el lenguaje de especificación formal XFL3 que ofrecen interfaces de usuario gráficas para facilitar el flujo del diseño en las etapas de descripción, ajuste, verificación y síntesis. El lenguaje denominado XFL3, facilita la construcción de reglas complejas expresadas lingüísticamente permitiendo el uso de factores de corrección en las reglas, así como cualquier clase de función conectiva que relacione los antecedentes y etiquetas lingüísticas que puedan ser aplicados a antecedentes simples o conectados. Adicionalmente, este lenguaje permite la inclusión de nuevos operadores definidos por el propio usuario y la definición de sistemas modulares jerárquicos. El flujo de diseño con XFuzzy podría ilustrarse de la siguiente manera:



A continuación pasamos a describir las herramientas CAD, así como las funcionalidades que cada una de ellas aporta:

(A) **xfedit**: facilita la descripción de la estructura lógica de un sistema difuso, conformado por sus entradas, salidas, grupos de funciones de pertenencia para cada variable, conjunto de operadores para cada regla, reglas base, y el sistema de arquitectura (como se interconectan las reglas).

(B) **xfpkg**: facilita la definición de paquetes de funciones, es decir, los bloques de código que describen los parámetros, expresiones matemáticas y otras propiedades de las funciones de pertenencia, así como métodos de defuzzificación, y funciones unarias y binarias relacionadas respectivamente con el lenguaje y los conectores difusos.

(C) **xf2dplot** y **xf3dplot**: visualizan gráficamente una de las salidas del sistema conjuntamente con 2 o 3 de sus entradas.

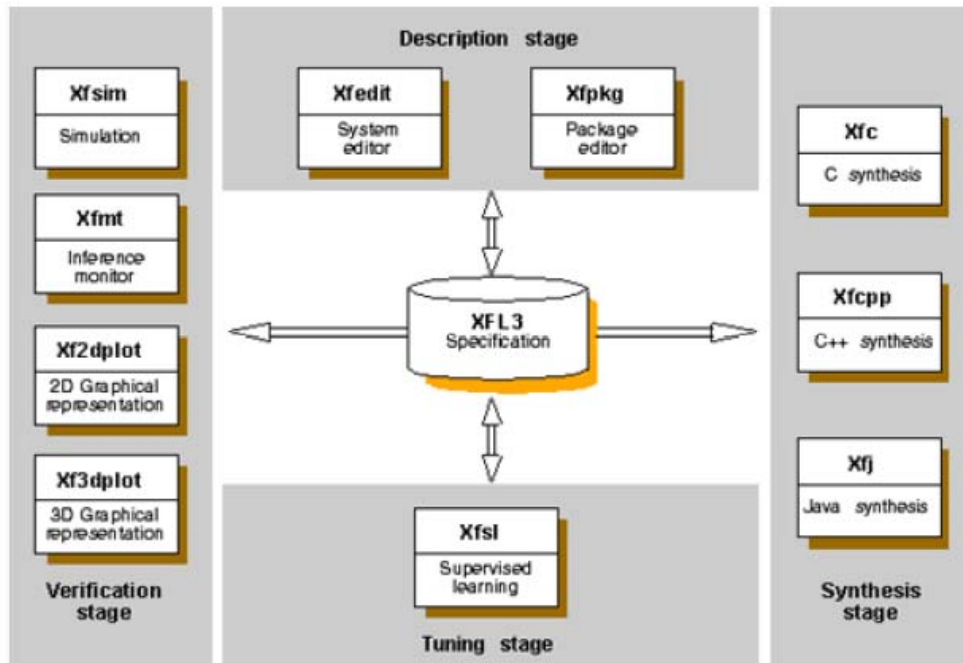
(D) **xfmt**: monitoriza como los valores de salida se obtienen en la interacción con los valores de entrada.

(E) **xfsim**: simula cómo se comporta el sistema difuso en el dominio de la aplicación.

(F) **xfls**: permite la aplicación de un amplio conjunto de algoritmos de aprendizaje supervisados (de segundo orden, gradiente-descendentes, de Gauss-Newton y más algoritmos estadísticos) a cualquier sistema definido en XFL3.

(G) **xfc**, **xfcc** y **xfji**: los cuales respectivamente traducen un sistema descrito en XFL3 a los lenguajes de programación C, C++ y Java. Esta última funcionalidad aunque sin representar la panacea definitiva, proporciona una ayuda que ha resultado clave para la elección de la herramienta en nuestro proyecto.

A continuación podemos ver un gráfico esquemático de lo comentado durante este apartado:



Por último, como conclusión podemos decir que:



El núcleo del entorno está formado por un conjunto de funciones comunes denominado librería XFL. Los elementos de dicha librería realizan la translación y el análisis semántico de las especificaciones XFL, y las almacenan.

Los módulos que facilitan las diferentes etapas de diseño se agrupan en torno a esta librería, utilizando sus servicios. Alrededor de estos módulos, el entorno dispone de

una interfaz gráfica de usuario que proporciona un método de acceso intuitivo y simple a sus elementos.

4. Sistema de Valoración Borrosa de Inmuebles

4.1. Descripción Funcional

La función de nuestra aplicación, como ha quedado expuesto a lo largo de las secciones anteriores, es proporcionar una valoración comercial sobre un inmueble que gestione un filtro de información ante el potencial cliente, así como de forma indirecta ayudar al mismo a elegir el tipo de inmueble que más se adapte a sus preferencias en función de una serie de características de referencia del inmueble, ofreciendo al comprador una lista ordenada por la valoración comercial de los inmuebles.

Nuestro sistema tendrá como variables de entrada el tipo de comprador, además de una serie de características del inmueble (estado del portal, el número de habitaciones, el número de baños...) y decidirá el inmueble que mejor valoración tiene basándose en unas reglas ya definidas con anterioridad.

Cada inmueble se valora de acuerdo a características comunes habitualmente utilizadas por promotores inmobiliarios y agentes. Estas características son:

- ✓ Tipo: Piso, pareado, adosado, apartamento, estudio o chalet independiente.

El tipo de vivienda se considera de especial interés para posibles clasificaciones de las muestras y es una variable muy dependiente del tipo de cliente que se divide en cuatro grupos.

- ✓ Luminosidad: Grado de luminosidad (1 - 10).

La luminosidad es una variable muy valorada por algunas personas, fundamentalmente por mujeres y más si son madres de familia. Es una variable que se incrementa al incrementarse la edad de la persona.

- ✓ Orientación: Norte, Sur, Este, Oeste o combinaciones de ambas.
Puede influir en la preferencia del cliente cuando puede elegir entre dos productos extremadamente parecidos, pero no es relevante en primera instancia.
- ✓ Representatividad: Grado de representatividad (0 - 10).
Es muy valorada por cualquier comprador. En este caso se refiere a la representatividad del interior.
- ✓ Estado del portal: Grado de estado del portal (0 - 10).
Relacionada con la representatividad, es fundamental para el cliente. A veces es más importante que el propio inmueble, ya que la modificación de las zonas comunes es más complicada.
- ✓ Fachada: Grado de fachada (0 - 10)
Le pasa lo mismo que al estado del portal pero debe estudiarse por separado.
- ✓ Vistas: Grado de satisfacción de lo que se ve desde las ventanas del inmueble.
- ✓ Número de Habitaciones: Se trata de una variable nítida que puede ser determinante para el cliente.
- ✓ Número de Baños: Al igual que el número de habitaciones, es una variable nítida.
- ✓ Altura sobre rasante: Planta en la que se encuentra el inmueble. En principio es un dato nítido.
- ✓ Ascensor: Un dato muy importante es la accesibilidad al inmueble, dato que medimos mediante la existencia o no de ascensor.
- ✓ Antigüedad del inmueble: Relacionada con la representatividad. Es importante debido a que puede ser motivo de gastos imprevistos y problemas posteriores al cierre de la operación.

- ✓ Valoración de Zonas Comunes: Son datos de mucha importancia a la hora de elegir un inmueble.
- ✓ Estado General del Inmueble: Grado del estado (0 - 10)
Supone gastos en reformas, por lo que se debe tener muy en cuenta a la hora de realizar la valoración.
- ✓ Metros Construidos: Es importante porque el cliente representará el valor del patrimonio que adquiere.
- ✓ Metros Habitables: Es similar a lo que ocurre con los metros construidos.
- ✓ Precio de Tasación: Puede ser un dato utilizable para la valoración.
- ✓ Precio de Salida: Dato fundamental para la valoración del vendedor.
- ✓ Precio de Venta: Dato fundamental para la valoración del vendedor y de la empresa gestora.
- ✓ Semanas en oferta: Fundamental para refinar la valoración del inmueble. Se añade valor si es pequeño y se decrementa si es grande.
- ✓ Valoración del experto: Es un dato difuso que representa un valor general que la percepción del experto le hace entender sobre el inmueble.

Como se observa, las variables descritas poseen distinto grado de dependencia entre ellas. En este trabajo no las hemos tenido en cuenta por simplicidad. La valoración difusa de cada característica se ha considerado suficiente al estar implícita la dependencia o independencia. Las posibles contingencias derivadas de este asunto también damos por hecho que están suavizadas o no son representativas dado que la descripción y clasificación viene realizada por un experto.

Con el fin de realizar una valoración por sectores poblacionales, se ha solicitado la ayuda de un experto que ha puesto al descubierto cuatro grupos representativos:

- ❖ Pareja joven sin hijos (φ_1)
- ❖ Familia de 2 – 3 hijos (φ_2)
- ❖ Hombre soltero de clase media-alta (φ_3)
- ❖ Mujer soltera de unos 40 años y cierta solvencia (φ_4)

Puesto que los expertos coinciden en establecer estos cuatro posibles tipos de compradores, damos por hecho que podremos clasificar a cualquier comprador en uno de estos cuatro grupos sin pérdida de generalidad (en caso contrario rechazaríamos al individuo por dato atípico).

Cada comprador tiene una serie de propiedades que le caracterizan a la hora de elegir un inmueble. Esta clasificación del sector poblacional previa a la valoración del producto ha dado buenos resultados como puede verse en publicaciones relacionadas. Un autor referencia en este campo es Tim Harford [Financial Times, Harford].

En primer lugar el experto ha concluido la existencia de unas características primarias que representan criterios de decisión que en primera instancia motivan al comprador. Son las siguientes:

- ❖ Pareja joven sin hijos \Rightarrow Zona centro, 2 dormitorios, plaza de garaje y precio aproximado de 240.000 euros.
- ❖ Familia de 2 – 3 hijos \Rightarrow 3-4 dormitorios, plaza de garaje, zonas comunes, preferiblemente urbanización y luminoso.
- ❖ Hombre soltero \Rightarrow Apartamento, plaza de garaje, lo más céntrico posible y con un precio aproximado de 300.000 euros.
- ❖ Mujer soltera \Rightarrow 2 dormitorios, luminoso, con ascensor, a ser posible con plaza de garaje y con un precio aproximado a 180.000 euros.

Sin embargo, además de éstas características primarias debemos tener también en cuenta una serie de características secundarias. El mismo comprador refina su decisión, lo que se traduce en una búsqueda más sesgada. En esta etapa se evalúan las llamadas características secundarias. Así pues, después de hacer comprobado que un subconjunto de inmuebles cumple las características primarias, evaluaremos las secundarias con el fin de ofrecer al comprador un subconjunto adecuado formado por una serie de inmuebles sobre el cual el comprador pueda tomar una decisión adecuada.

La siguiente tabla, mostrada en la página siguiente, refleja la importancia que se le da a cada elemento desde tres puntos de vista diferentes: Prioritario para el comprador, secundario para el comprador, tasador. Estos datos se valoran con tres posibles grados: 0 (nada importante), 1 (algo importante) y 2 (muy importante).

Respecto a la aplicación de la evaluación, consideraremos por tanto los siguientes datos de entrada:

- Un perfil $i = 1, 2, 3$ o 4 correspondiente al perfil del comprador φ_i
- C = Valor del inmueble respecto a los elementos fundamentales para el comprador.
- P = Valor del inmueble respecto a los elementos particulares para el comprador.
- T = Valor del inmueble respecto a los elementos fundamentales para la tasación.

	Elemento	φ_1			φ_2			φ_3			φ_4		
		P	S	T	P	S	T	P	S	T	P	S	T
1	Tipo de Inmueble	2	2	2	2	2	2	2	2	2	2	2	2
2	Luminosidad	0	0	2	2	2	2	0	1	2	2	2	2
3	Orientación	0	0	2	0	2	2	0	1	2	0	2	2
4	Representatividad	0	2	2	2	2	2	0	2	2	2	1	2
5	Estado del Portal	0	1	2	2	2	2	0	2	2	0	1	2
6	Fachada	0	2	1	2	1	1	0	2	1	0	1	1
7	Vistas	0	2	1	1	2	1	0	1	1	2	2	1
8	Nº de Habitaciones	0	2	1	2	2	1	0	1	1	2	2	1
9	Nº de Baños	0	2	1	2	2	1	0	1	1	0	1	1
10	Antigüedad	0	0	1	1	2	1	0	1	1	0	1	1
11	Altura (planta)	0	1	1	0	2	1	0	1	1	0	2	1

12	Ascensor	0	0	2	1	2	2	1	2	2	1	2	2
13	Plazas de Garaje	0	1	2	1	2	2	0	2	2	0	2	2
14	Estado General Inmueble	0	1	1	1	1	1	0	2	1	0	2	1
15	Zonas Comunes	1	1	1	2	2	1	2	1	1	2	1	1
16	Zona y Subzona	1	2	2	2	1	2	2	2	2	2	2	2
17	Metros Construidos	1	2	2	1	0	2	2	0	2	2	0	2
18	Metros Habitables	1	2	1	2	2	1	2	1	1	2	2	1
19	Precio de Tasación	Np	0	Np	Np	0	Np	Np	0	Np	Np	0	Np
20	Precio de Salida	2	2	Np	2	2	Np	0	2	Np	0	2	Np
21	Precio de Venta	2	2	Np	2	2	Np	0	2	Np	0	2	Np
22	Semanas en Oferta	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np
23	Valoración experto	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np
24	Observaciones	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np	Np

Una ordenación de preferencias asociada a cada perfil sobre el conjunto de características puede observarse en las siguientes figuras:

Pareja joven sin hijos φ_1

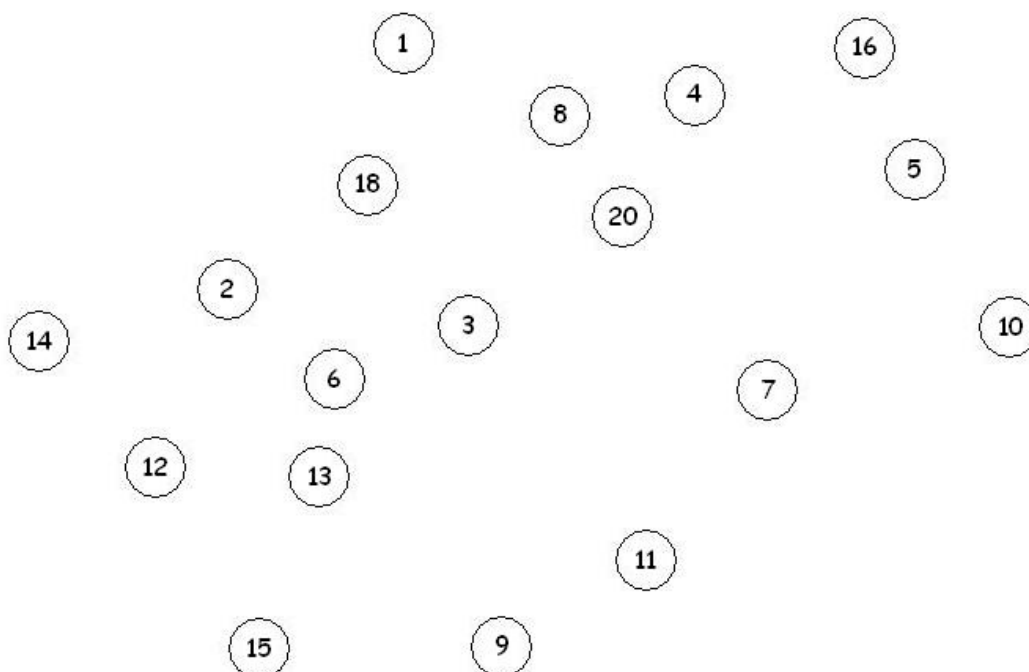


Figura 1

Familia de 2 – 3 hijos φ_2

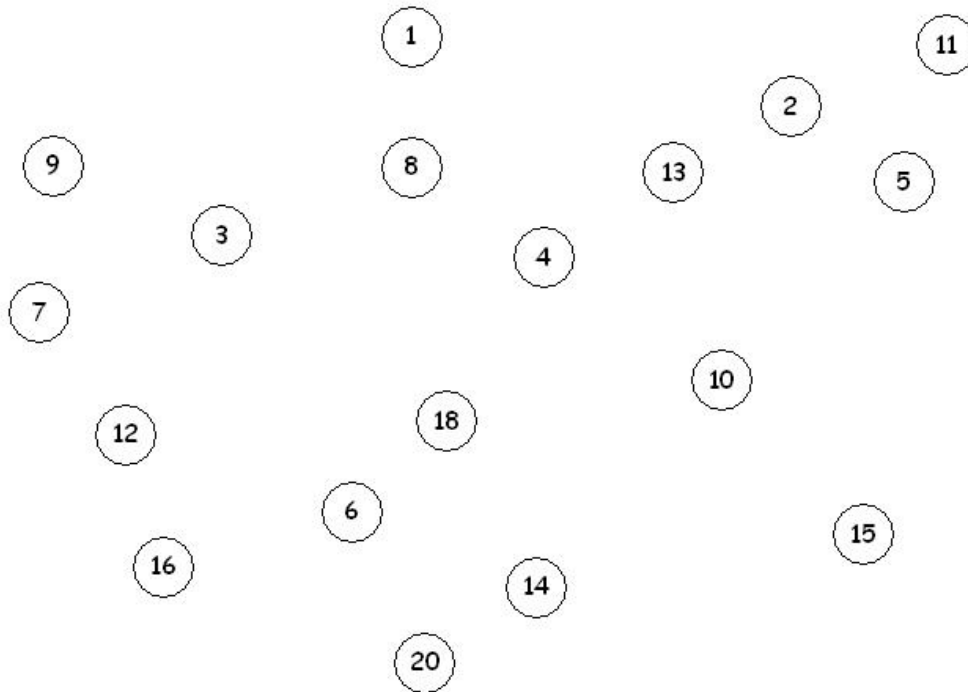


Figura 2

Hombre soltero de clase media-alta φ_3

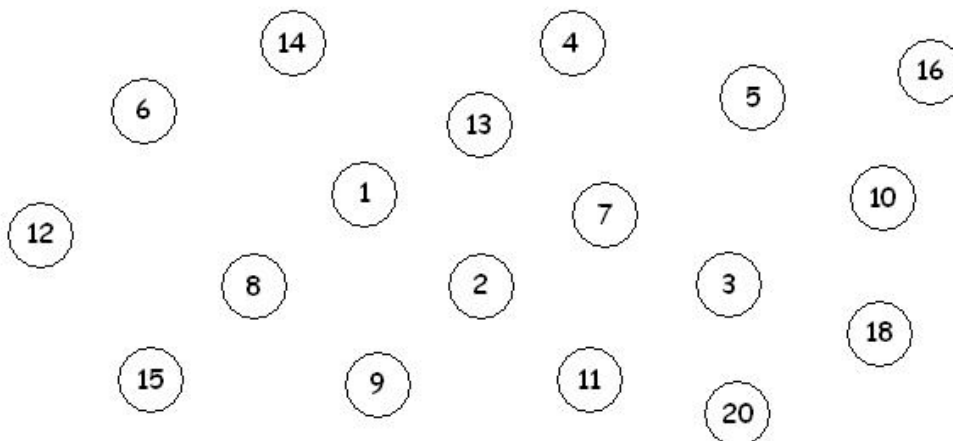


Figura 3

Mujer Soltera de unos 40 años y cierta solvencia φ_4

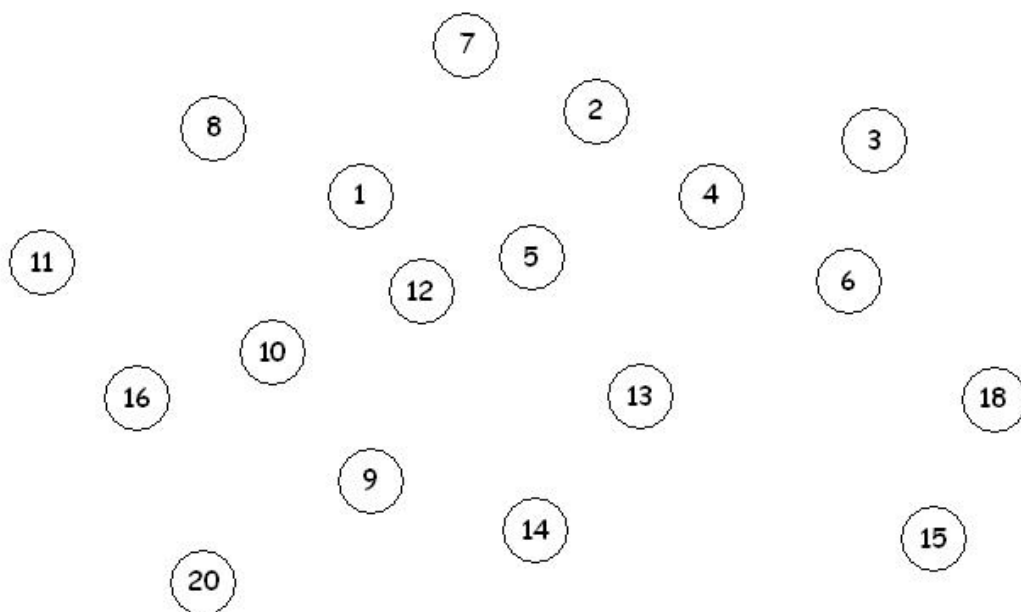


Figura 4

Los datos recogidos en este apartado conforman una especificación funcional robusta sobre la que trabajar en el desarrollo de la aplicación. La labor de construcción de esta especificación, como se ha venido comentando, viene en gran parte realizada por un experto en la materia que participa de la iniciativa, aunque el equipo del proyecto ha participado en tareas de refinamiento y adaptación de la misma a la problemática a la que nos enfrentamos.

Cálculo de la valoración de los inmuebles

Para realizar el cálculo de la valoración de un inmueble se realiza una media ponderada de todos los elementos del inmueble.

Se denomina media (aritmética) ponderada de un conjunto de números al resultado de multiplicar cada uno de los números por un valor particular para cada uno de ellos, llamado su peso, obteniendo a continuación la suma de estos productos, y dividiendo el resultado de esta suma de productos entre la suma de los pesos más la

masa según la característica de cada número inicial. Este "peso" depende de la importancia o significancia de cada uno de los valores. O dicho de otro modo es un promedio en el que cada valor de observación se pondera con algún índice de su importancia

Para una serie de datos

$$X = \{x_1, x_2, \dots, x_n\}$$

a la que corresponden los pesos

$$W = \{w_1, w_2, \dots, w_n\}$$

La media ponderada se calcula como:

$$\bar{x} = \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i}$$

o:

$$\bar{x} = \frac{x_1 w_1 + x_2 w_2 + x_3 w_3 + \dots + x_n w_n}{w_1 + w_2 + w_3 + \dots + w_n}$$

Para el cálculo de la valoración de los inmuebles se ha utilizado la valoración obtenida por el sistema X-Fuzzy (x_n) ponderándola con el peso de la importancia correspondiente del elemento (w_n).

Este cálculo se realiza dos veces para cada inmueble: un primer cálculo con las importancias del usuario como pesos, y una segunda vez utilizando las importancias del experto, obteniendo la valoración del usuario y valoración del experto por separado.

Una vez obtenidos ambos cálculos, se obtiene la valoración general del inmueble realizando una media aritmética de los dos valores:

$$\text{Valoración General} = \frac{\text{Valoración Usuario} + \text{Valoración Experto}}{2}$$

El cálculo empleado encaja por apropiado y por tener la dificultad apropiada en relación al alcance de nuestro proyecto, no obstante en trabajos posteriores podrían considerarse otros tipos de agregación más sofisticados.

4.2. Desarrollo del Sistema

4.2.1. Definición del sistema

A grandes rasgos, la construcción y desarrollo del sistema pasa por la integración de la descripción del mismo realizada en la herramienta X-fuzzy dentro de un proyecto Java que lo interconecta con el resto de funcionalidades de la aplicación. En las tres secciones siguientes se describen las tres fases en las que se ha dividido el desarrollo de la aplicación:

En primer lugar tenemos la fase que consiste en el “Desarrollo del Motor de Inferencia”: Antes de comenzar el desarrollo de la aplicación es necesario realizar una descripción funcional completa del modelo en el sistema CAD X-Fuzzy. Se realiza la definición de tipos del sistema, tanto de entrada como de salida, y las funciones de inferencia correspondientes.

Una vez realizado el sistema de inferencia es necesario modelar el resto de la aplicación para integrar correctamente el sistema generado por X-Fuzzy. La integración se compone de dos partes:

- Desarrollo de la Base de Datos: Es necesario el diseño de una base de datos para almacenar todos los inmuebles para así almacenar la información y tener una fuente de entrada y salida de datos.
- Integración de la aplicación: Este apartado corresponde a la implementación de la aplicación de usuario final.

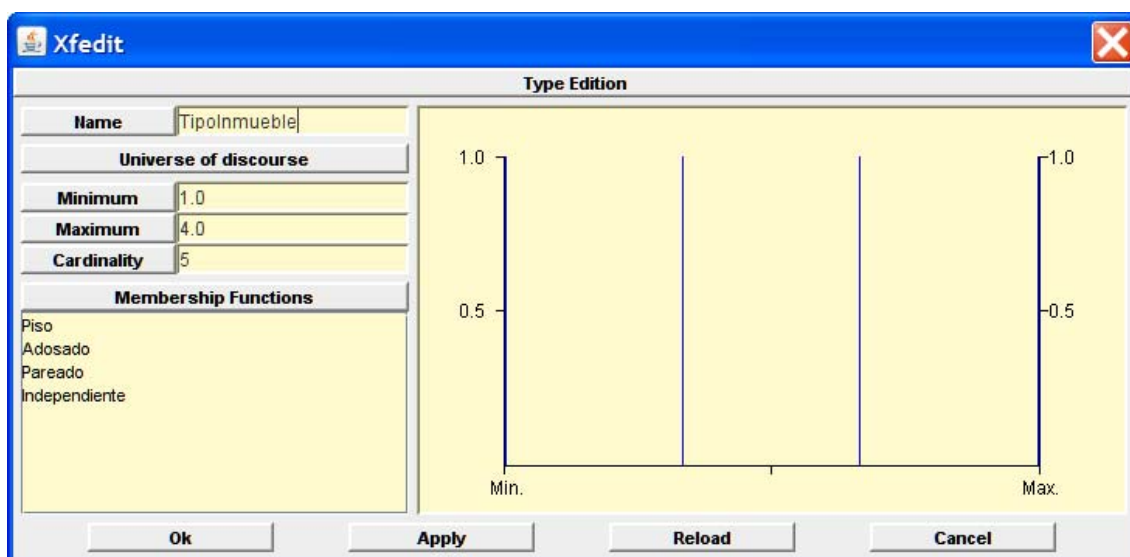
4.2.2. Desarrollo del Motor de Inferencia

Como se ha descrito anteriormente, es necesario modelar el sistema de inferencia en la herramienta CAD X-Fuzzy.

En primer lugar se han de describir todos los tipos de datos que se van a utilizar. Hay que tener en cuenta que los tipos son siempre difusos. A continuación mostramos algunos de los tipos modelados:

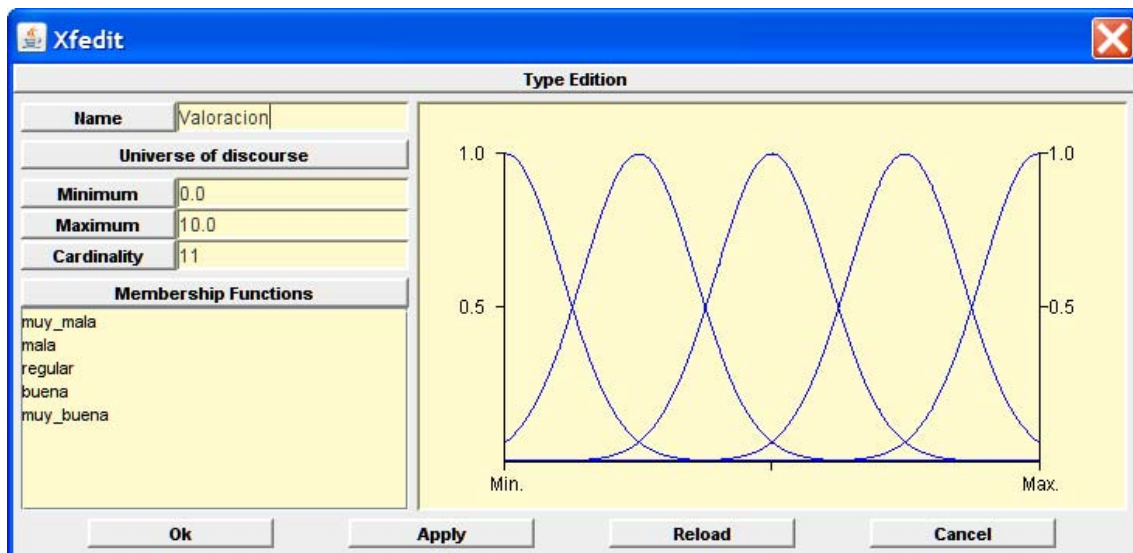
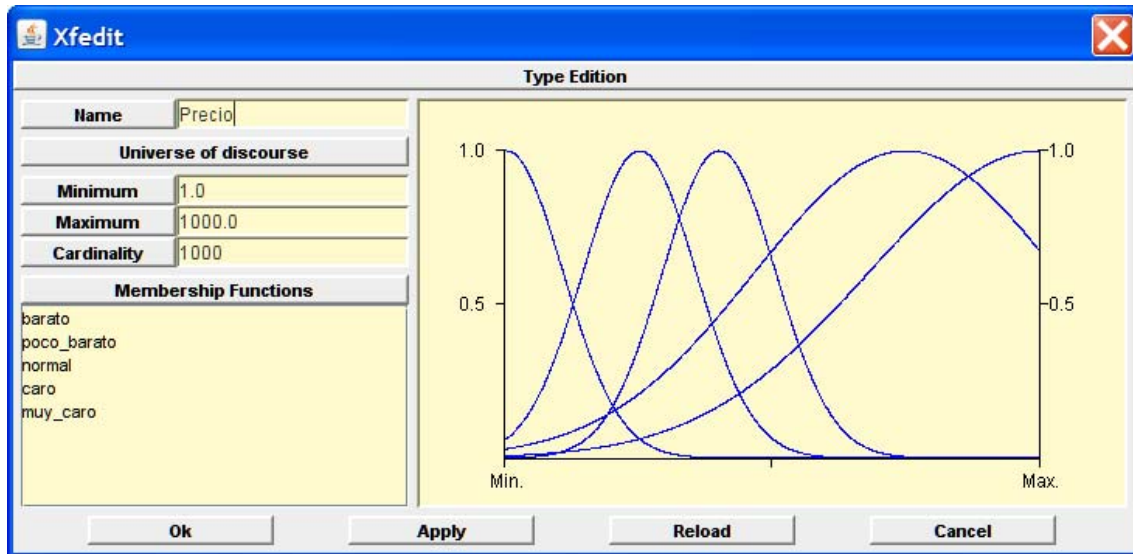
- Tipos singleton:

Este tipo es necesario cuando los datos son claramente discretos, como por ejemplo TipoInmueble:



- Tipos con componente difusa:

A continuación se muestran unos ejemplos de tipos de datos que por su naturaleza deben transcribirse mediante procesos difusos en función de las características de referencia. Estos son *Precio* y *Valoración*.



Una vez representados los tipos, se han de modelar las funciones de inferencia. Para crear una función de inferencia se deben especificar qué datos de entrada y salida tiene y sus correspondientes tipos, en este momento tendremos una “caja negra”

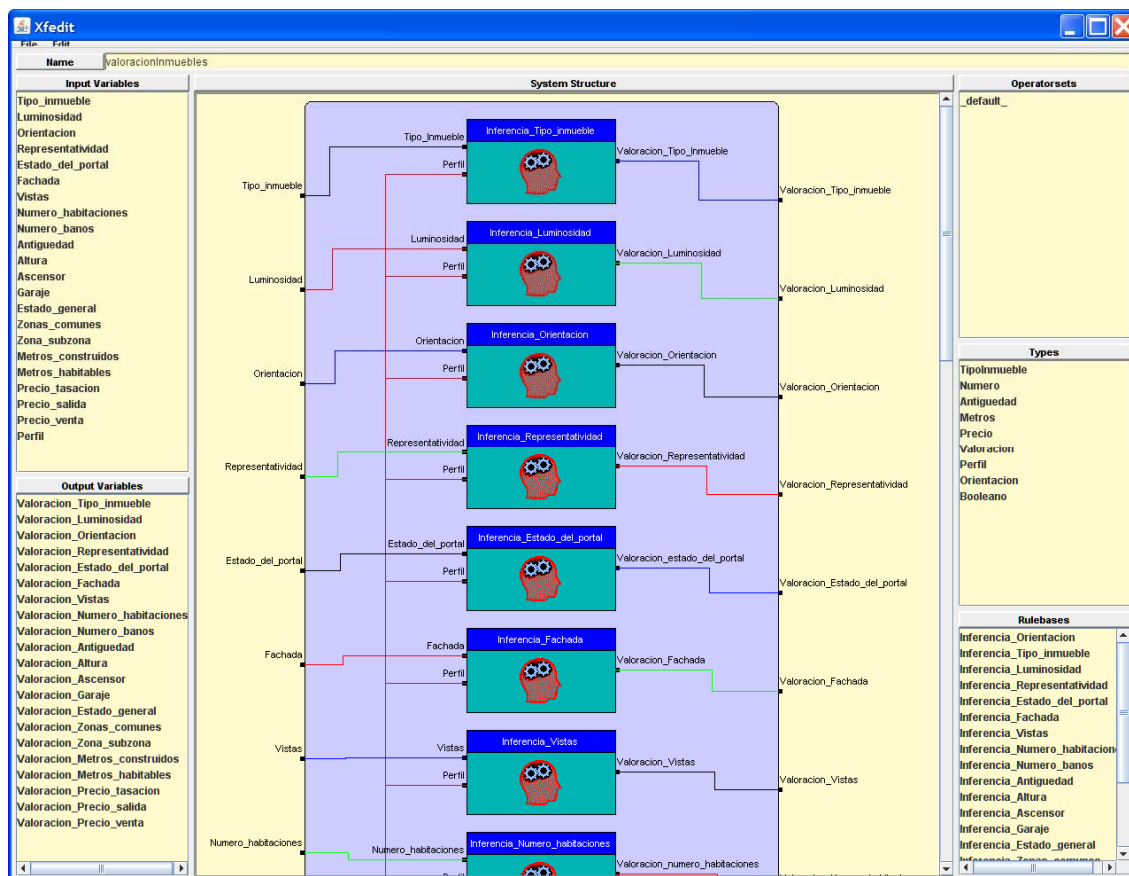
Tras la creación de la caja negra se ha de proceder a rellenar las posibles combinaciones de los distintos tipos de valores que pueden tener los datos de entrada para así obtener los de salida. Esto se realiza mediante el editor de reglas que usa X-Fuzzy.

En la siguiente figura se muestra el editor de reglas de inferencia mediante el cual se introducen los predicados y se asocian a variables y operadores, conformando las reglas. Como puede verse siguen el siguiente esquema:

Si (Característica == valor & Perfil == perfil) -> valoración_caracteristica = valoración

Rule	Premise	Conclusion
0	if (Luminosidad == muy_mala & Perfil == Pareja_joven_sin_hijos)	-> Valoracion_Luminosidad = muy_mala
1	if (Luminosidad == muy_mala & Perfil == Pareja_joven_sin_hijos)	-> Valoracion_Luminosidad = regular
2	if (Luminosidad == muy_mala & Perfil == Matrimonio_2_3_hijos)	-> Valoracion_Luminosidad = muy_mala
3	if (Luminosidad == muy_mala & Perfil == Hombre_soltero)	-> Valoracion_Luminosidad = mala
4	if (Luminosidad == muy_mala & Perfil == Mujer_soltera)	-> Valoracion_Luminosidad = muy_mala
5	if (Luminosidad == mala & Perfil == Pareja_joven_sin_hijos)	-> Valoracion_Luminosidad = regular
6	if (Luminosidad == mala & Perfil == Matrimonio_2_3_hijos)	-> Valoracion_Luminosidad = mala
7	if (Luminosidad == mala & Perfil == Hombre_soltero)	-> Valoracion_Luminosidad = regular
8	if (Luminosidad == mala & Perfil == Mujer_soltera)	-> Valoracion_Luminosidad = mala
9	if (Luminosidad == regular & Perfil == Pareja_joven_sin_hijos)	-> Valoracion_Luminosidad = regular
10	if (Luminosidad == regular & Perfil == Matrimonio_2_3_hijos)	-> Valoracion_Luminosidad = regular
11	if (Luminosidad == regular & Perfil == Hombre_soltero)	-> Valoracion_Luminosidad = regular
12	if (Luminosidad == regular & Perfil == Mujer_soltera)	-> Valoracion_Luminosidad = regular
13	if (Luminosidad == buena & Perfil == Pareja_joven_sin_hijos)	-> Valoracion_Luminosidad = regular
14	if (Luminosidad == buena & Perfil == Matrimonio_2_3_hijos)	-> Valoracion_Luminosidad = buena
15	if (Luminosidad == buena & Perfil == Hombre_soltero)	-> Valoracion_Luminosidad = buena
16	if (Luminosidad == buena & Perfil == Mujer_soltera)	-> Valoracion_Luminosidad = buena
17	if (Luminosidad == muy_buena & Perfil == Pareja_joven_sin_hijos)	-> Valoracion_Luminosidad = regular
18	if (Luminosidad == muy_buena & Perfil == Matrimonio_2_3_hijos)	-> Valoracion_Luminosidad = muy_buena
19	if (Luminosidad == muy_buena & Perfil == Hombre_soltero)	-> Valoracion_Luminosidad = buena
20	if (Luminosidad == muy_buena & Perfil == Mujer_soltera)	-> Valoracion_Luminosidad = muy_buena
*		

Por último, se relacionan todas las funciones de inferencia necesarias para obtener el sistema final. En la siguiente figura se muestra un esquema del sistema, con sus “cajas” de inferencia y variables de entrada y salida.



4.2.3. Desarrollo de la Base de Datos

La capacidad para acceder a una Base de Datos desde Java la ofrece la API JDBC (Java DataBase Connectivity). JDBC es un estándar para manejar bases de datos en Java. ODBC es un estándar de Windows para manejar bases de datos, de forma que cualquier programa en Windows que desee acceder a bases de datos genéricas debe usar este estándar.

Para almacenar las características de los diferentes inmuebles de los que la aplicación hace uso disponemos de una Base de Datos creada en Microsoft Access desde la que accedemos a través de nuestra aplicación.

La Base de Datos está formada por tres tablas llamadas *Viviendas*, *TipoInmueble* y *Orientacion*.

La tabla *Viviendas* es en donde guardamos los atributos que poseen los distintos inmuebles (clasificados según el tipo al que pertenecen):

Tipo Texto:

- DMXX
- Tipo de inmueble
- Orientación
- Zona / Subzona
- Dirección

Tipo Número:

- Luminosidad
- Representatividad
- Estado del Portal
- Estado General
- Fachada
- Vistas
- N° Dormitorios
- N° Baños / Aseos
- Antigüedad
- Planta / Altura
- Plazas de garaje
- Metros Construidos
- Metros Habitables

Tipo Booleano (Sí / No):

- Ascensor
- Zonas Comunes
- Amueblado
- Piscina
- Aire Acondicionado

- Conserje
- Trastero

Tipo Monetario:

- Precio de Tasación
- Precio de Salida
- Precio de Venta

Tiene como clave principal al atributo DMXX.

En ésta tabla tenemos almacenados 300 inmuebles para valorarlos según el perfil seleccionado.

La tabla *TipoInmueble* contiene una única columna llamada Tipo con los cuatro tipos de inmuebles que puede poseer la base de datos:

- | | |
|------------|-------------------------|
| a) Piso | b) Adosado |
| c) Pareado | d) Chalet Independiente |
| e) Estudio | f) Apartamento |

Ésta columna es a la vez clave principal de la tabla.

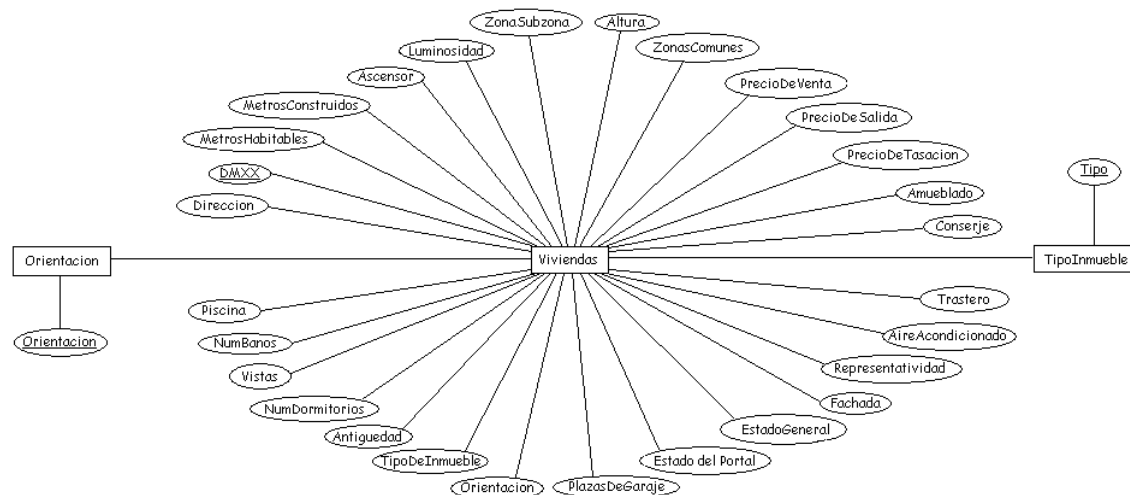
La tabla *Orientación*, al igual que la tabla *TipoInmueble*, está formada por una única columna que es a su vez la clave principal. Ésta columna contiene ocho filas, cada una con una posible orientación que podría tener el inmueble. Estos son:

- Norte
- Sur
- Este
- Oeste
- Noreste
- Noroeste
- Sureste
- Suroeste

Estas tres tablas están relacionadas entre sí, de tal forma que en el atributo Tipo de Inmueble de la tabla Viviendas sólo podrá aparecer un valor que esté contenido en la

columna Tipo de la tabla TipoInmueble. Y en el atributo Orientación de Viviendas sólo encontraremos un valor contenido en la tabla Orientación.

Veamos el Diseño del Modelo Relacional:



El diseño está normalizado hasta la Tercera Forma Normal. Para que se cumpla la Primera Forma Normal se tiene que dar que los dominios de los atributos sólo puedan ser atómicos, para evitar así atributos multivalorados, compuestos y sus combinaciones. Como podemos observar en el diseño, todos los atributos cumplen que son atómicos, luego podemos afirmar que está en Primera Forma Normal.

La Segunda Forma Normal también la cumple ya que para cada atributo que no forma parte de la clave primaria depende funcional y completamente de cada clave.

Por último se cumple la Tercera Forma Normal, ya que se satisface la Segunda Forma Normal y ninguno de los atributos que no forman parte de la clave primaria depende transitivamente de la clave primaria.

A continuación veremos cómo utilizar la base de datos:

Para conectar la Base de Datos con nuestra aplicación lo primero que debemos hacer es hacer una llamada al Driver JDBC-ODBC para cargarlo. Para ello usamos las siguientes líneas de código:

```
try {  
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
} catch(Exception e) {  
System.out.println("No se ha podido cargar el Driver JDBC-ODBC");  
}
```

Con esto ya tenemos cargado el driver. Ahora básicamente trabajamos con tres objetos. Estos objetos son: *Connection*, *Statement* y *ResultSet*. El objeto *Connection* se obtiene al realizar la conexión a la Base de Datos. El objeto *Statement* se crea a partir del anterior y nos permite ejecutar SQL para hacer consultas o modificaciones en la Base de Datos. En caso de hacer una consulta (SELECT ... FROM ...) se nos devolverá un objeto que representa los datos que deseamos consultar; este objeto es un objeto *ResultSet* (Hoja de resultados).

Una vez cargado el driver debemos realizar la conexión a nuestra Base de Datos. Para ello usamos el objeto *Connection* de la siguiente forma:

```
Connection conexion = DriverManager.getConnection("jdbc:odbc:MS Access  
Database;DBQ=BBDD.mdb", "Nombre_Usuario", "Contraseña");
```

Por ultimo, una vez conectada la Base de Datos con la aplicación ya podremos hacer todas las consultas que deseemos. Para ello utilizaremos los objetos *Statement* y *ResultSet*. A continuación mostramos un ejemplo de cómo funcionan, en este ejemplo lo que buscamos es obtener todos los datos de los inmuebles que contiene la tabla *Viviendas* ordenado por su identificador (DMXX):

```
String consulta = "SELECT * FROM Viviendas ORDER BY DMXX";  
Statement stat = conexion.createStatement();  
ResultSet resultado = stat.executeQuery(consulta);
```

4.2.4. Desarrollo de la Aplicación

Esta parte implementa un interfaz software con un controlador borroso que permite visualizar los resultados obtenidos de evaluar los diferentes inmuebles para cada posible tipo de controlador.

Para implementar el sistema hemos creado un proyecto en el entorno Eclipse con el JDK 1.6. Como detalles de implementación, destacar que la parte de lógica fuzzy es realizada por las clases Java generadas por la herramienta XFuzzy.

4.2.4.1. Patrón de Diseño Modelo-Vista-Controlador

Introducción Modelo-Vista-Controlador (MVC)

Modelo-Vista-Controlador (MVC) es un patrón de diseño que separa la interfaz de una aplicación, de los datos de la misma y de la lógica de control.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada como un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que debe ser aplicable a diferentes problemas de diseño en distintas circunstancias.

En cualquier programa que hagamos con este patrón podemos encontrar tres partes bien diferenciadas:

- Por un lado tenemos el problema que tratamos de resolver. Este problema suele ser independiente de cómo queramos que nuestro programa recoja los resultados o cómo queremos que los presente. Este código constituiría el **modelo**.

- Otra parte clara es la presentación visual que queramos hacer del juego. Esta parte del código es la **vista**. La llamaré interfaz gráfica por ser lo más común, pero podría ser de texto, de comunicaciones con otro programa externo, con la impresora, etc.
- La tercera parte de código es aquel código que toma decisiones, algoritmos, etc. Es código que no tiene que ver con las ventanas visuales ni con las reglas de nuestro modelo. Esta parte del código es el **controlador**.

El patrón MVC se suele utilizar principalmente en aplicaciones Web, ya que en ellas es frecuente cambiar la vista manteniendo la lógica del sistema y el control de flujo. Además en este tipo de desarrollos, suele ocurrir que las personas que se encargan de la vista no tengan conocimientos muy amplios de programación y con éste patrón se pueden encargar sólo de la vista sin preocuparse de lo demás.

Ventajas e Inconvenientes del patrón MVC

El patrón MVC tiene una serie de ventajas:

- Hay una clara separación entre sus elementos ya que cada uno está especializado en su tarea. La vista muestra los datos al usuario, el controlador toma las decisiones y el modelo trata el problema a resolver.
- Es muy sencillo el obtener distintas representaciones para un mismo modelo, esto genera una mayor facilidad para el desarrollo en múltiples dispositivos ya que podemos variar la vista según sus capacidades.
- Es posible construir nuevas vistas sin necesidad de modificar el modelo debido a la reutilización de sus componentes.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos.
- Es fácil desarrollar prototipos rápidos.
- Se produce una mayor claridad de diseño.
- Se facilita el mantenimiento.

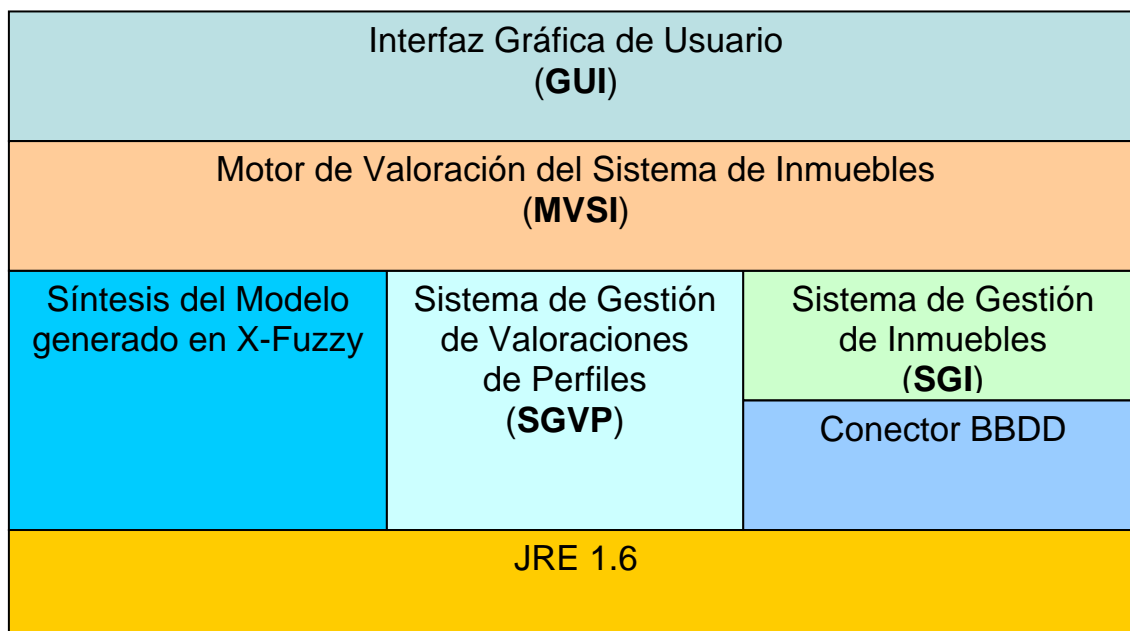
- Los prototipos desarrollados suelen ser más escalables.

Pero no todos son ventajas, también tiene algún que otro inconveniente, como son:

- Tiene que ceñirse a una estructura predefinida, lo que a veces puede incrementar la complejidad del sistema. Hay problemas que son más difíciles de resolver respetando el patrón MVC.
- La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

4.2.4.2. Arquitectura del sistema

A continuación presentamos el esquema jerárquico de la arquitectura del sistema representada en módulos funcionales:



El proyecto está desarrollado bajo la plataforma Java 1.6. En la parte baja de la arquitectura se observa JRE (Java Runtime Environment), que es la parte correspondiente

a la Máquina Virtual de Java, que es la encargada de ejecutar el proyecto sobre el sistema operativo.

En el nivel inmediatamente superior observamos los siguientes módulos:

- Síntesis del Modelo en X-Fuzzy: Consiste en todos los módulos de la aplicación generados por el entorno CAD X-Fuzzy. Contiene todos los tipos de datos, funciones de inferencia y arquitectura del sistema modelado en X-Fuzzy. El módulo por si sólo no tiene posee ninguna funcionalidad, es necesario integrarlo en un sistema para utilizar las funciones de inferencia para representar y analizar los resultados;
- Sistema de Gestión de Valoraciones de Perfiles (SGVP): Este módulo se encarga de gestionar cada valoración e importancia que un perfil determinado da a cada elemento;
- Conector BBDD: Interfaz que permite la conexión con la BBDD. El conector se ha implementado para utilizar una base de datos creada en Access, pero con ligeras modificaciones se puede adaptar a cualquier otro gestor de base de datos;
- Sistema de Gestión de Inmuebles (SGI): Este módulo contiene toda la información correspondiente al modelo de los inmuebles (identificar del inmueble, características, localización...). Utiliza el módulo “Conector BBDD” para cargar en la aplicación todos los inmuebles almacenados en la base de datos;

En la capa superior, nos encontramos con el módulo “Motor de Valoración del Sistema de Inmuebles” (MVSI). Este módulo es núcleo de la aplicación, utiliza todos los módulos de la capa inferior para generar los datos de salida y para proporcionárselos a la capa superior, correspondiente a la interfaz de usuario.

La interfaz gráfica de usuario (GUI), se encarga de intermediar entre el motor de la aplicación y el usuario, permitiendo la entrada de datos por parte del usuario y proporcionando los datos de salida del motor de la aplicación (MVSI), todo ello gráficamente.

4.2.4.3. Relación Arquitectura - Implementación

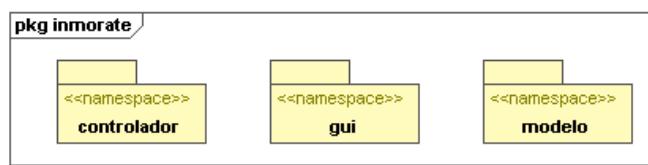
El proyecto está adaptado al patrón MVC. A continuación se describe la relación entre los módulos y los paquetes de la implementación:

Package	Módulo correspondiente	Descripción
controlador	MVSI	Encargado de ejecutar la aplicación y relacionar el GUI con el motor de inferencia.
Vista	GUI	Interfaz gráfico de usuario.
modelo		Todos los paquetes correspondiente al modelo de aplicación
modelo.bbdd	Conector de BBDD	Conexión a la base de datos.
modelo.constants	SGVP y SGI	Todas las constantes de ajuste (importancias, valoraciones, perfiles, tipos de usuario...)
modelo.core	MVSI	Motor de la aplicación que llama a las funciones de X-Fuzzy y realiza los cálculos de salida.
modelo.elementos	SGI	Representa una característica de un inmueble (tipo, tamaño, situación...)
modelo.inmueble	SGI	Clases para la representación de los inmuebles.
modelo.sector	SGVP	Características de cada sector poblacional. Se encarga de gestionar los tipos de perfiles existentes y las importancias y valoraciones que da el sector

		correspondiente a cada elemento del inmueble.
modelo.valoración	SGVP	Elementos que contienen las valoraciones. Son los valores de salida de las funciones de inferencia.
modelo.xfuzzy	Síntesis del modelo generado en X-Fuzzy	Motor de inferencia sintetizado por X-Fuzzy.

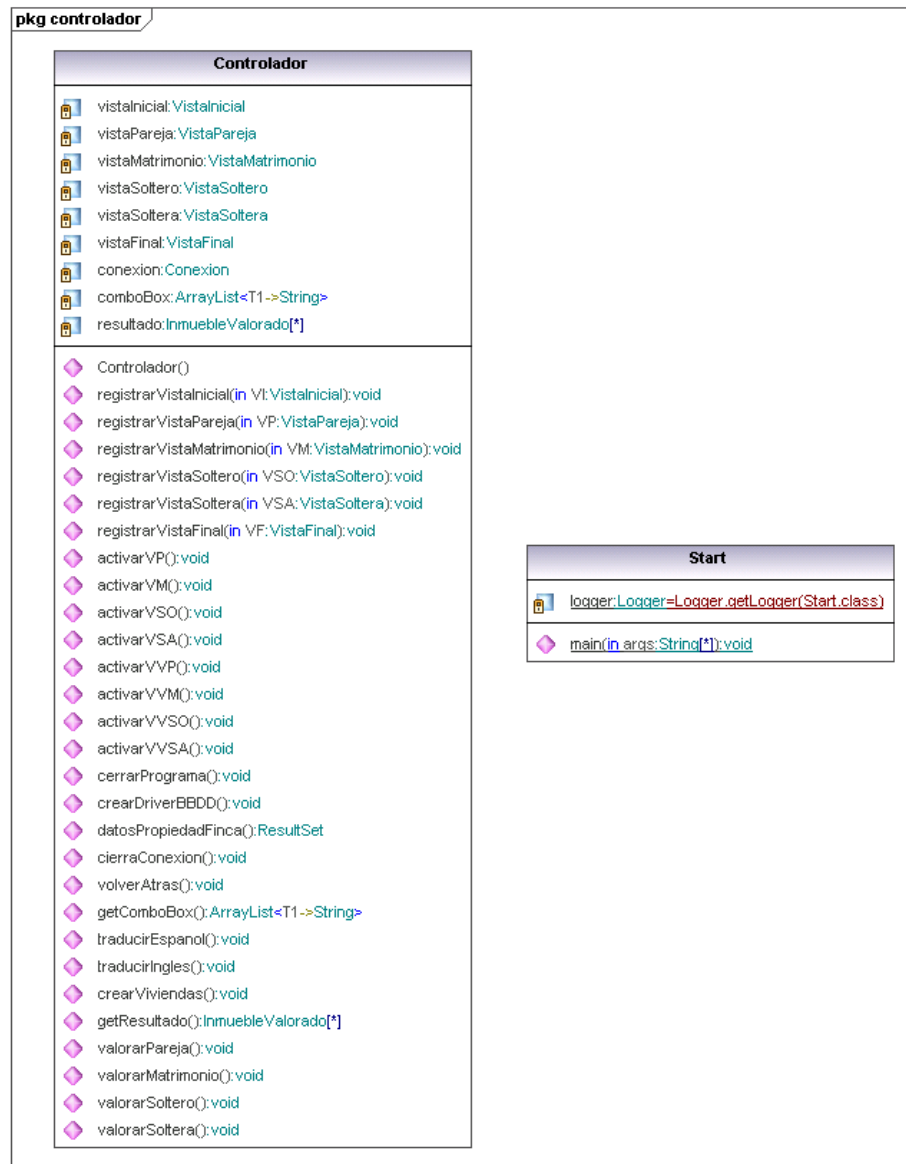
4.2.4.4. Diagrama de paquetes de la aplicación

El sistema está formado por los siguientes paquetes:



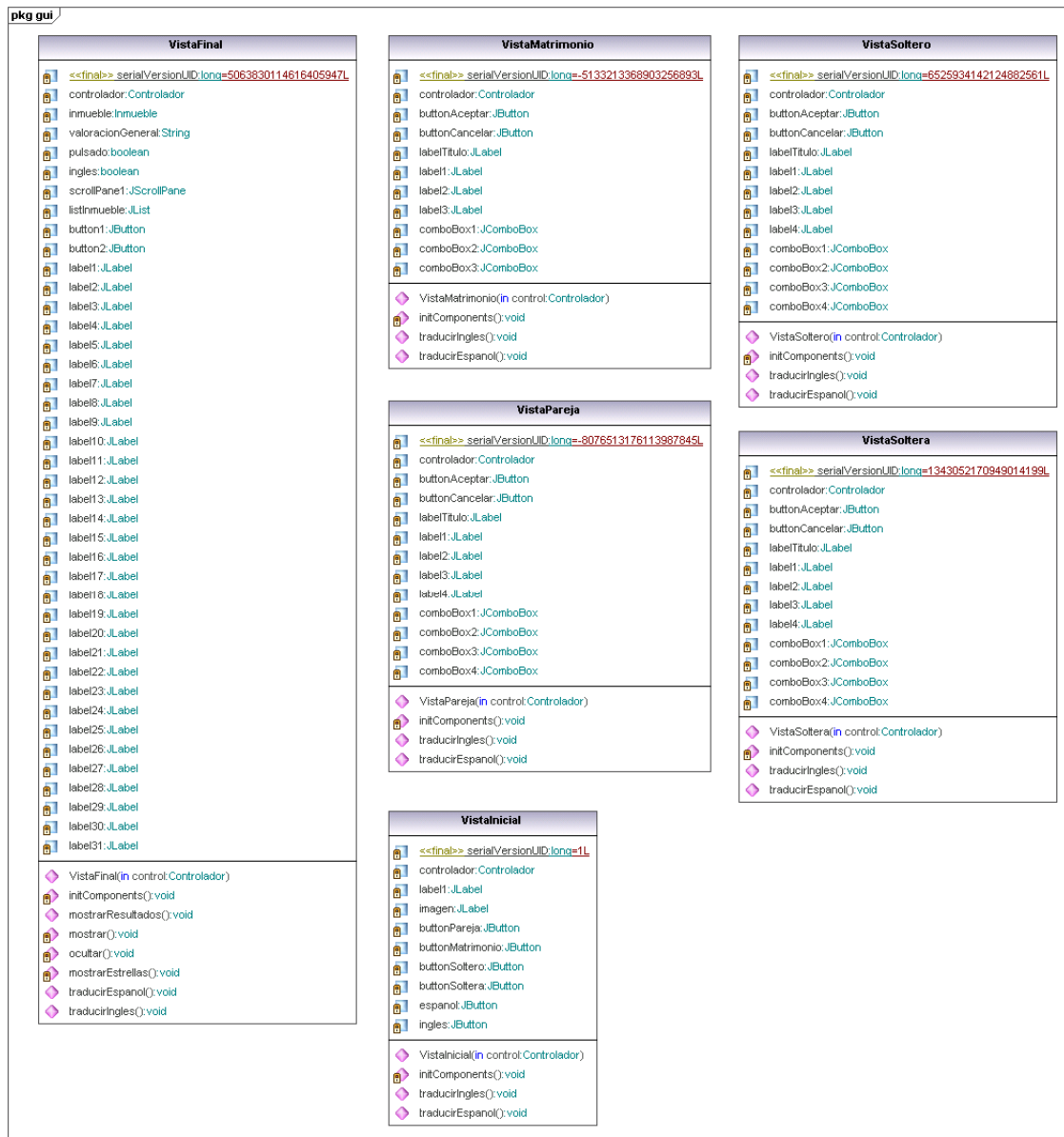
Paquete controlador:

El paquete *controlador*, como su nombre indica, agrupa clases que gobiernan las conexiones y el correcto funcionamiento de la aplicación.



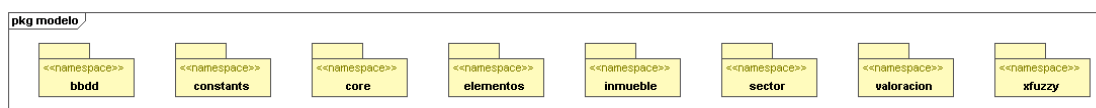
Paquete gui:

El paquete *gui* agrupa todas las clases que definen las funcionalidades gráficas de la aplicación. A continuación podemos ver su estructura:



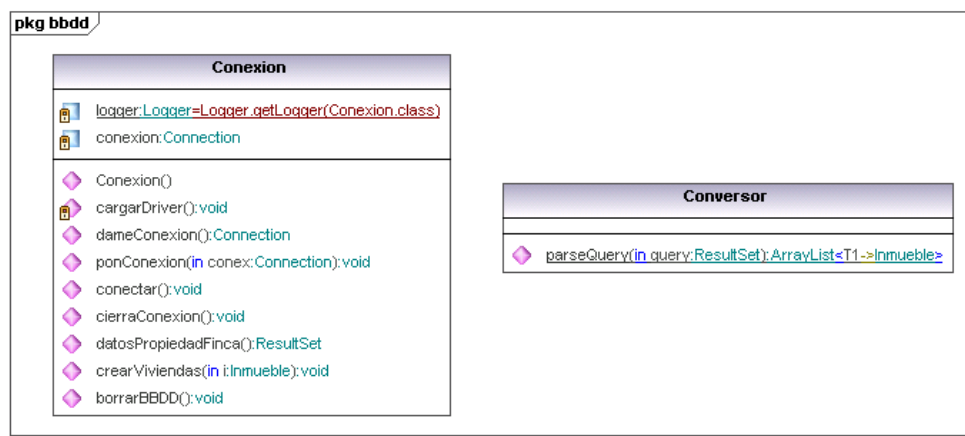
Paquete modelo:

En este paquete se agrupa la parte de la implementación correspondiente a la definición del modelo de la aplicación, dividida en los siguientes sub-paquetes de los cuales se aportó una definición en el apartado 4.2.1:



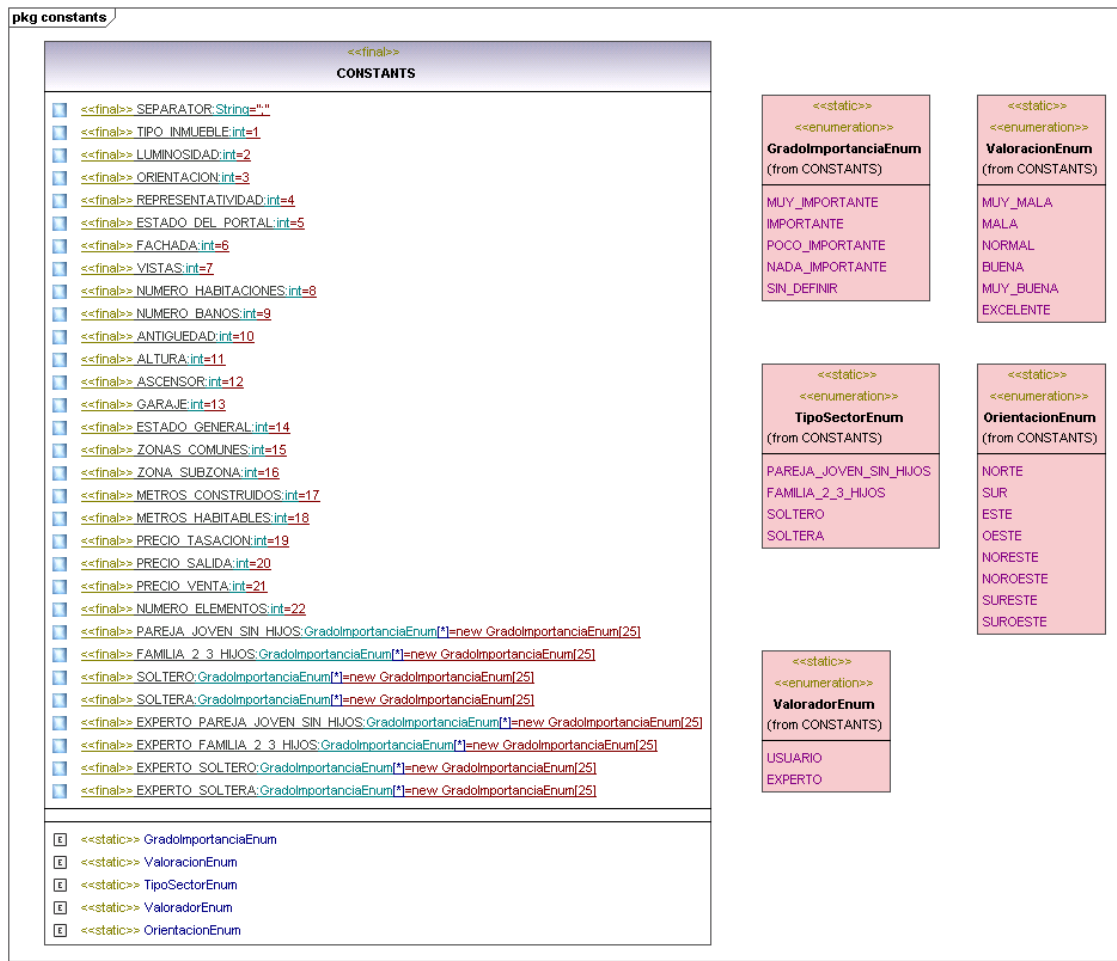
Paquete modelo.bbdt

En el siguiente paquete, se implementa la interfaz para la conexión de la aplicación con la base de datos de inmuebles.



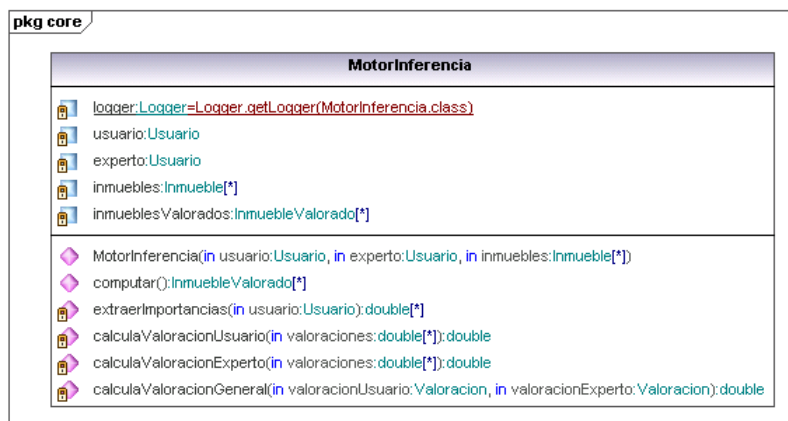
Paquete modelo.constants

Paquete de definición de las constantes de ajuste que actuarán como conjunto de características de referencia.



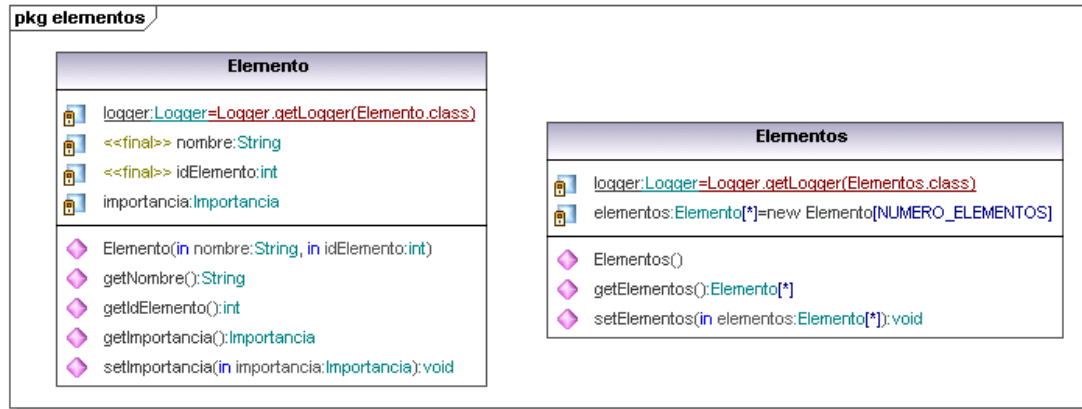
Paquete modelo.core

Engloba las clases encargadas de gestionar el motor de inferencia.



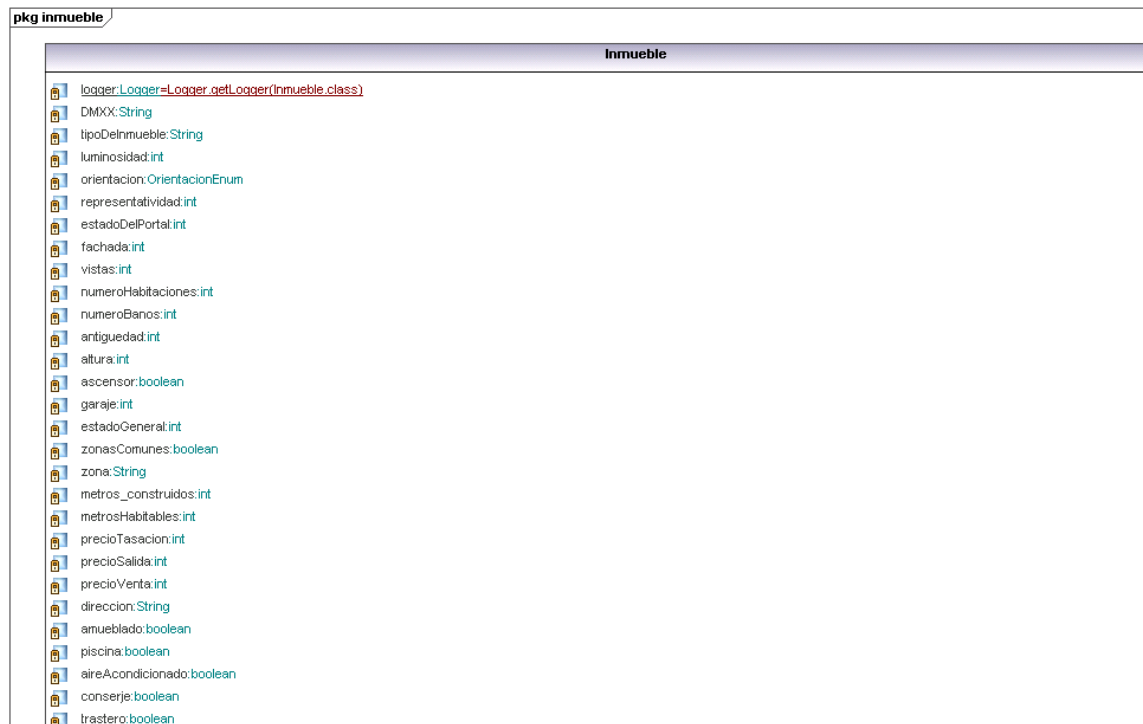
Paquete modelo.elementos

Agrupar las clases necesarias para cada elemento o característica que conforma un inmueble.



Paquete modelo.inmueble






















Clases involucradas en la generación de inmuebles. Para su mejor visualización, presentamos dividida la clase inmueble en atributos y métodos, y luego por separado la clase InmuebleValorado.



```

◆ Inmuelle()
◆ Inmuelle(in dmxx:String, in tipoDeInmueble:String, in luminosidad:int, in orientacion:OrientacionEnum, in representatividad:int, in estadoDelPortal:int, in fachada:int, in vista
◆ isAireAcondicionado():boolean
◆ setAireAcondicionado(in aireAcondicionado:boolean):void
◆ isAmueblado():boolean
◆ setAmueblado(in amueblado:boolean):void
◆ getAntiguedad():int
◆ setAntiguedad(in antiguedad:int):void
◆ isAscensor():boolean
◆ setAscensor(in ascensor:boolean):void
◆ isConserje():boolean
◆ setConserje(in conserje:boolean):void
◆ getDireccion():String
◆ setDireccion(in direccion:String):void
◆ getDMXX():String
◆ setDMXX(in dmxx:String):void
◆ getEstadoDelPortal():int
◆ setEstadoDelPortal(in estadoDelPortal:int):void
◆ getEstadoGeneral():int
◆ setEstadoGeneral(in estadoGeneral:int):void
◆ getFachada():int
◆ setFachada(in fachada:int):void
◆ getLuminosidad():int
◆ setLuminosidad(in luminosidad:int):void
◆ getMetrosConstruidos():int
◆ setMetrosConstruidos(in metrosConstruidos:int):void
◆ getMetrosHabitables():int
◆ setMetrosHabitables(in metrosHabitables:int):void
◆ getNumeroBanos():int
◆ setNumeroBanos(in numAseos:int):void
◆ getNumeroHabitaciones():int
◆ setNumeroHabitaciones(in numDormitorios:int):void
◆ getOrientacionAsString():String
◆ getOrientacion():OrientacionEnum
◆ setOrientacion(in orientacion:OrientacionEnum):void
◆ getGaraje():int
◆ setGaraje(in garaje:int):void
◆ getMetros_construidos():int
◆ setMetros_construidos(in metros_construidos:int):void
◆ setOrientacion(in orientacion:String):void
◆ isPiscina():boolean
◆ setPiscina(in piscina:boolean):void
◆ getAltura():int
◆ setAltura(in altura:int):void
◆ getPlazasGaraje():int
◆ setPlazasGaraje(in plazasGaraje:int):void
◆ getPrecioSalida():int
◆ setPrecioSalida(in precioSalida:int):void
◆ getPrecioTasacion():int
◆ setPrecioTasacion(in precioTasacion:int):void
◆ getPrecioVenta():int
◆ setPrecioVenta(in precioVenta:int):void
◆ getRepresentatividad():int
◆ setRepresentatividad(in representatividad:int):void
◆ getTipoDeInmueble():String
◆ setTipoDeInmueble(in tipoDeInmueble:String):void
◆ isTrastero():boolean
◆ setTrastero(in trastero:boolean):void
◆ getVistas():int
◆ setVistas(in vistas:int):void
◆ getZona():String
◆ setZona(in zona:String):void
◆ isZonasComunes():boolean
◆ setZonasComunes(in zonasComunes:boolean):void
◆ crearInmueble(in valor:int):void
◆ extraerDatos():double[*]
◆ defuzzyTipoInmueble():double
◆ defuzzyOrientacion():double
◆ defuzzyAscensor():double
◆ defuzzyZonasComunes():double
◆ defuzzyZonaSubzona():double










```

InmuebleValorado	
 <code>logger:Logger=Logger.getLogger(InmuebleValorado.class)</code>  <code>inmueble:Inmueble</code>  <code>valoraciones:Valoracion[]</code>  <code>valoracionGeneral:Valoracion</code>  <code>valoracionUsuario:Valoracion</code>  <code>valoracionExperto:Valoracion</code>	
 <code>InmuebleValorado(In inmueble:Inmueble, In valoraciones:Valoracion[], In valoracionGeneral:Valoracion, In valoracionUsuario:Valoracion, In valoracionExperto:Valoracion)</code>  <code><<annotations>> toString():String</code>  <code>toStringOnlyValues():String</code>  <code>toStringOnlyValuesFuzzy():String</code>  <code>getInmueble():Inmueble</code>  <code>setInmueble(In inmueble:Inmueble):void</code>  <code>getValoraciones():Valoracion[]</code>  <code>setValoraciones(In valoraciones:Valoracion[]):void</code>  <code>getValoracionGeneral():Valoracion</code>  <code>setValoracionGeneral(In valoracionGeneral:Valoracion):void</code>  <code>getValoracionUsuario():Valoracion</code>  <code>setValoracionUsuario(In valoracionUsuario:Valoracion):void</code>  <code>getValoracionExperto():Valoracion</code>  <code>setValoracionExperto(In valoracionExperto:Valoracion):void</code>  <code>toFile(In filename:String, In inmueblesValorados:InmuebleValorado[]):void</code>  <code>toFile(In filename:String, In inmueblesValorados:InmuebleValorado[], In fuzzy:boolean):void</code>	

Paquete modelo.sector

Clase en la que se define el sector poblacional (soltero, soltera, pareja...)

También existe la distinción entre usuario y experto.

Sector	
 <code>logger:Logger=Logger.getLogger(Sector.class)</code>  <code><<final>> tipoSector:TipoSectorEnum</code>	
 <code>Sector(In tipoSector:TipoSectorEnum)</code>  <code>toString():String</code>  <code>getAsString(In idSector:TipoSectorEnum):String</code>  <code>getAsString(In idSector:int):String</code>  <code>getId():int</code>  <code>getId(In idSector:TipoSectorEnum):int</code>  <code>getTipoSector():TipoSectorEnum</code>	

Paquete modelo.valoracion

Clases resultantes y dependientes del motor de inferencia.

<p>Usuario</p> <pre> logger:Logger=Logger.getLogger(Usuario.class) <<final>> sector:Sector elementos:Elementos valorador:ValoradorEnum Usuario(in sector:Sector, in valorador:ValoradorEnum) inicializaImportanciaElementos():void setImportanciaTipoInmueble(in importancia:Importancia):void setImportanciaLuminosidad(in importancia:Importancia):void setImportanciaOrientacion(in importancia:Importancia):void setImportanciaRepresentatividad(in importancia:Importancia):void setImportanciaEstadoDelPortal(in importancia:Importancia):void setImportanciaFachada(in importancia:Importancia):void setImportanciaVistas(in importancia:Importancia):void setImportanciaNumeroHabitaciones(in importancia:Importancia):void setImportanciaNumeroBanos(in importancia:Importancia):void setImportanciaAntiguedad(in importancia:Importancia):void setImportanciaAltura(in importancia:Importancia):void setImportanciaAscensor(in importancia:Importancia):void setImportanciaGaraje(in importancia:Importancia):void setImportanciaEstadoGeneral(in importancia:Importancia):void setImportanciaZonasComunes(in importancia:Importancia):void setImportanciaZonaSubzona(in importancia:Importancia):void setImportanciaMetrosConstruidos(in importancia:Importancia):void setImportanciaMetrosHabitables(in importancia:Importancia):void setImportanciaPrecioTasacion(in importancia:Importancia):void setImportanciaPrecioSalida(in importancia:Importancia):void setImportanciaPrecioVenta(in importancia:Importancia):void getImportanciaTipoInmueble():Importancia getImportanciaLuminosidad():Importancia getImportanciaOrientacion():Importancia getImportanciaRepresentatividad():Importancia getImportanciaEstadoDelPortal():Importancia getImportanciaFachada():Importancia getImportanciaVistas():Importancia getImportanciaNumeroHabitaciones():Importancia getImportanciaNumeroBanos():Importancia getImportanciaAntiguedad():Importancia getImportanciaAltura():Importancia getImportanciaAscensor():Importancia getImportanciaGaraje():Importancia getImportanciaEstadoGeneral():Importancia getImportanciaZonasComunes():Importancia getImportanciaZonaSubzona():Importancia getImportanciaMetrosConstruidos():Importancia getImportanciaMetrosHabitables():Importancia getImportanciaPrecioTasacion():Importancia getImportanciaPrecioSalida():Importancia getImportanciaPrecioVenta():Importancia getElementos():Elementos setElementos(in elementos:Elementos):void getSector():Sector </pre>	<p>IdentificadorValoracion</p> <pre> identificador:String valoracion:double posicion:int IdentificadorValoracion() getIdentificador():String setIdentificador(in identificador:String):void getValoracion():double setValoracion(in valoracion:double):void getPosicion():int setPosicion(in posicion:int):void </pre> <p>Valoracion</p> <pre> valor:double Valoracion() Valoracion(in valor:double) getValor():double setValor(in valor:int):void rangoMuyMala(in i:double):boolean rangoMala(in i:double):boolean rangoNormal(in i:double):boolean rangoBuena(in i:double):boolean rangoMuyBuena(in i:double):boolean rangoExcelente(in i:double):boolean fuzzy():ValoracionEnum fuzzy(in valor:double):ValoracionEnum defuzzy(in valoracion:ValoracionEnum):int </pre>	<p>Importancia</p> <pre> importancia:GradolImportanciaEnum Importancia(in importancia:GradolImportanciaEnum) getImportancia():GradolImportanciaEnum setImportancia(in importancia:GradolImportanciaEnum):void toString():String getAsString(in importancia:GradolImportanciaEnum):String getAsString(in importancia:int):String rangoNadainimportante(in i:double):boolean rangoPocoinimportante(in i:double):boolean rangoImportante(in i:double):boolean rangoMuyImportante(in i:double):boolean fuzzy(in val:Valoracion):ValoracionEnum fuzzy(in valor:double):ValoracionEnum defuzzy():int defuzzy(in importancia:GradolImportanciaEnum):int </pre>	<p>ValorarPareja</p> <pre> controlador:Controlador inmueblesValorados:InmuebleValorado[] comparisons:long=0 exchanges:long=0 ValorarPareja(in c:Controlador) valorar():InmuebleValorado[] maxPosiciones():InmuebleValorado[] exch(in a:InmuebleValorado[], in i:int, in j:int):void shuffle(in a:InmuebleValorado[]):void less(in x:double, in y:double):boolean quicksort(in a:InmuebleValorado[]):void quicksort(in a:InmuebleValorado[], in left:int, in right:int):void partition(in a:InmuebleValorado[], in left:int, in right:int):int </pre>	<p>ValorarMatrimonio</p> <pre> controlador:Controlador inmueblesValorados:InmuebleValorado[] comparisons:long=0 exchanges:long=0 ValorarMatrimonio(in c:Controlador) valorar():InmuebleValorado[] maxPosiciones():InmuebleValorado[] exch(in a:InmuebleValorado[], in i:int, in j:int):void shuffle(in a:InmuebleValorado[]):void less(in x:double, in y:double):boolean quicksort(in a:InmuebleValorado[]):void quicksort(in a:InmuebleValorado[], in left:int, in right:int):void partition(in a:InmuebleValorado[], in left:int, in right:int):int </pre>	<p>ValorarSoltera</p> <pre> controlador:Controlador inmueblesValorados:InmuebleValorado[] comparisons:long=0 exchanges:long=0 ValorarSoltera(in c:Controlador) valorar():InmuebleValorado[] maxPosiciones():InmuebleValorado[] exch(in a:InmuebleValorado[], in i:int, in j:int):void shuffle(in a:InmuebleValorado[]):void less(in x:double, in y:double):boolean quicksort(in a:InmuebleValorado[]):void quicksort(in a:InmuebleValorado[], in left:int, in right:int):void partition(in a:InmuebleValorado[], in left:int, in right:int):int </pre>	<p>ValorarSoltero</p> <pre> controlador:Controlador inmueblesValorados:InmuebleValorado[] comparisons:long=0 exchanges:long=0 ValorarSoltero(in c:Controlador) valorar():InmuebleValorado[] maxPosiciones():InmuebleValorado[] exch(in a:InmuebleValorado[], in i:int, in j:int):void shuffle(in a:InmuebleValorado[]):void less(in x:double, in y:double):boolean quicksort(in a:InmuebleValorado[]):void quicksort(in a:InmuebleValorado[], in left:int, in right:int):void partition(in a:InmuebleValorado[], in left:int, in right:int):int </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.3. Pruebas

El realizar distintas pruebas a nuestro sistema de valoración de inmuebles nos ha servido para ajustar determinados aspectos visuales de nuestra aplicación, facilitando así su manejo por parte del usuario.

Para mejorar la visualización de la aplicación y hacerla más atractiva de cara al usuario, optamos por mostrar la mínima cantidad posible de información acerca de los inmuebles.

Siempre que al usuario se le pida que decida sobre algún parámetro de la aplicación optamos por poner JComboBox. De ésta forma evitamos que el usuario pueda introducir valores erróneos, ya que tendrá que elegir entre un determinado rango de datos. Además añadimos un JSlider a la aplicación para que el usuario pueda poner límite al precio de salida del inmueble, que oscila entre 100.000 € y 500.000 €.

De una forma más comercial, se decidió que la aplicación estuviera traducida en dos idiomas: Español e Inglés.

4.3.1. Ejecución de las pruebas

Cuando ejecutamos la aplicación aparece en pantalla una ventana de bienvenida en la que se muestran los cuatro tipos de perfiles entre los que el usuario debe elegir al que pertenece (Pareja joven, Matrimonio, Soltero y Soltera)



Además de elegir el tipo de perfil de la persona que desea comprar el inmueble, en esta ventana de bienvenida tenemos la posibilidad de elegir el idioma en el que deseamos tener la aplicación.

Según el perfil seleccionado en la pantalla de bienvenida, la aplicación a continuación mostrará otra ventana en la que el usuario deberá elegir el grado de importancia de una serie de características propias del inmueble, así como el límite del precio de salida del inmueble que busca. Cada perfil tiene su propia ventana con unas características diferentes que debe valorar de muy importante a nada importante. En la parte de la derecha de la ventana tenemos los conjuntos difusos que representan las distintas clases de pertenencia para cada característica difusa del perfil elegido con anterioridad. Mientras en la parte de abajo se muestra la valoración general, la valoración del experto y la valoración del usuario para cada inmueble de la base de datos.

En el caso de la pareja joven sin hijos (ventana que se muestra a continuación), el usuario deberá valorar la importancia que tiene para él el tipo de inmueble, la representatividad, el estado del portal y los metros habitables.

Paraja Joven sin Hijos

Valore de mayor a menor importancia:

Tipo de Inmueble:

Representatividad:

Estado del Portal:

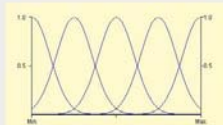
Metros Habitables:

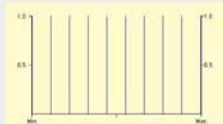
Límite de Precio:

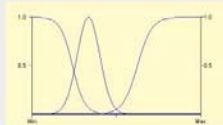
100.000 € 1.000.000 €


Xfuzzy

Busqueda basada en el perfil

Zona: 

Dormitorios: 

Plaza Garage: 

Precio: 

Para el matrimonio con hijos el usuario deberá valorar la importancia del estado del portal, el número de baños y la planta del inmueble. Ésta ventana tendrá la siguiente apariencia:

Matrimonio dos - tres hijos

Valore de mayor a menor importancia:

Estado del Portal:

Num. Baños / Aseos:

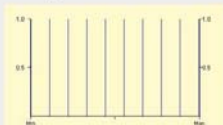
Altura (Planta):

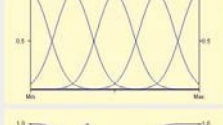
Límite de Precio:


100.000 € 1.000.000 €

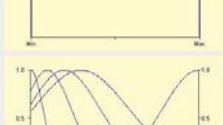
Xfuzzy


Busqueda basada en el perfil

Dormitorios: 

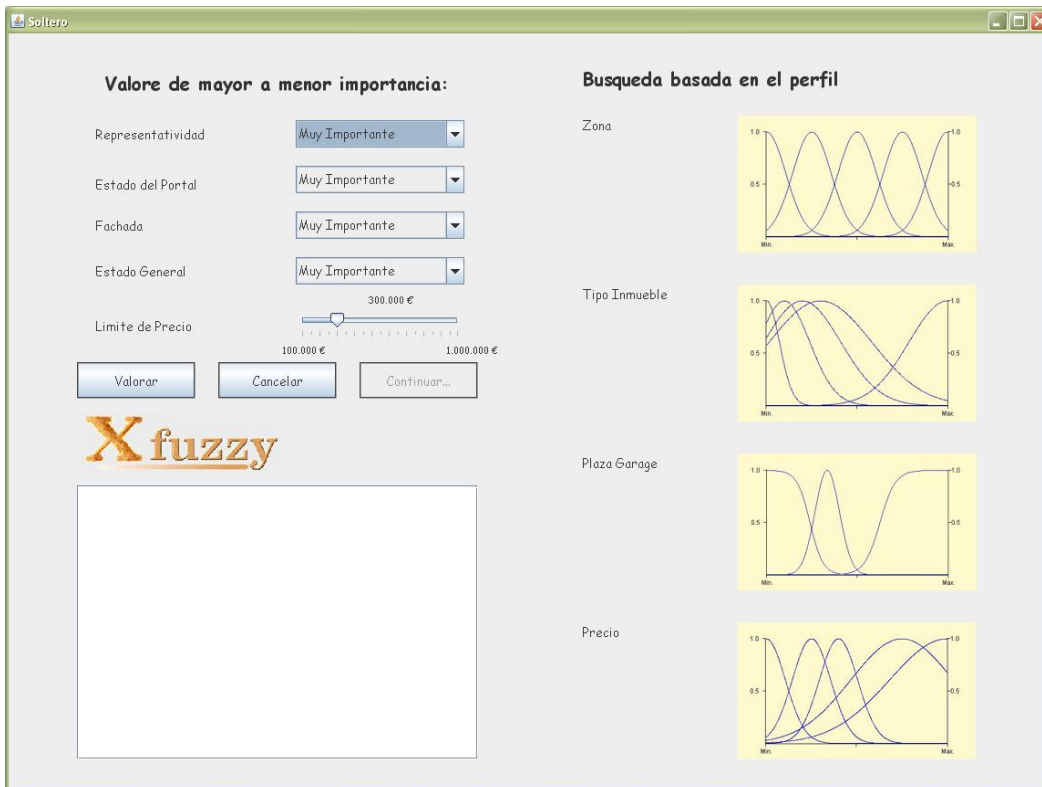
Luminosidad: 

Plaza Garage: 

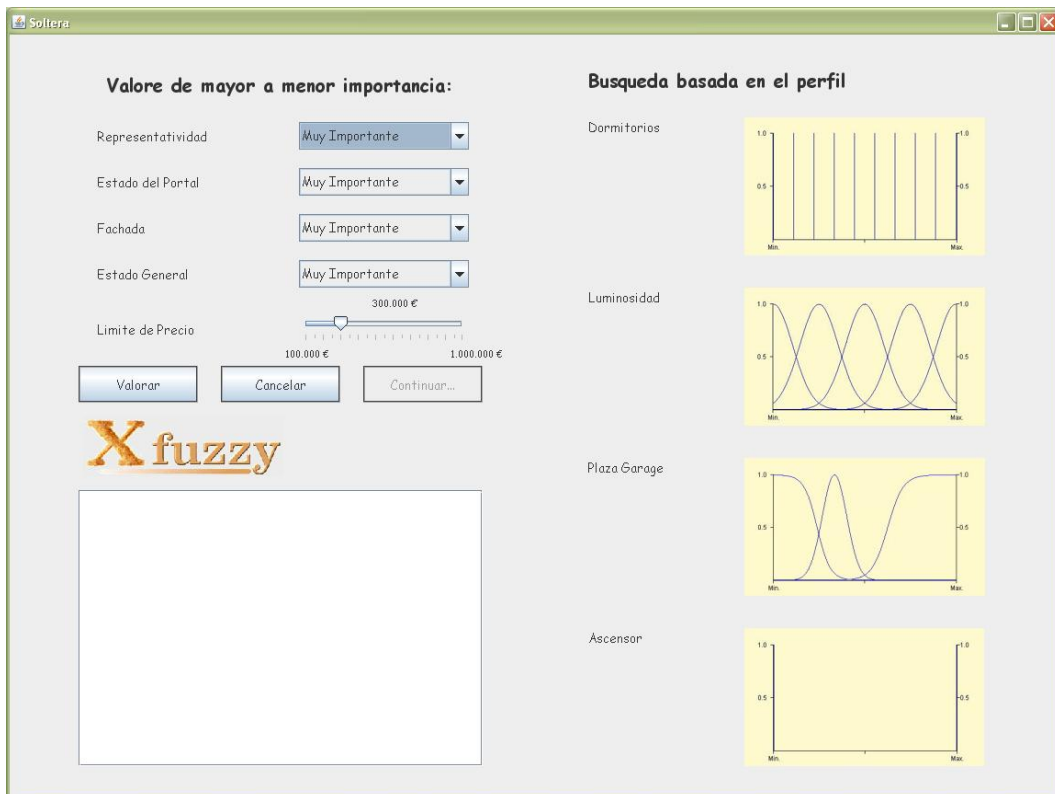
Zonas Comunes: 

Tipo Inmueble: 

La ventana del hombre soltero tiene cuatro características que el comprador tendrá que valorar: Representatividad, estado del portal, fachada y estado general. La apariencia de ésta ventana es:



Por último tenemos la ventana de la mujer soltera que tiene ésta apariencia:

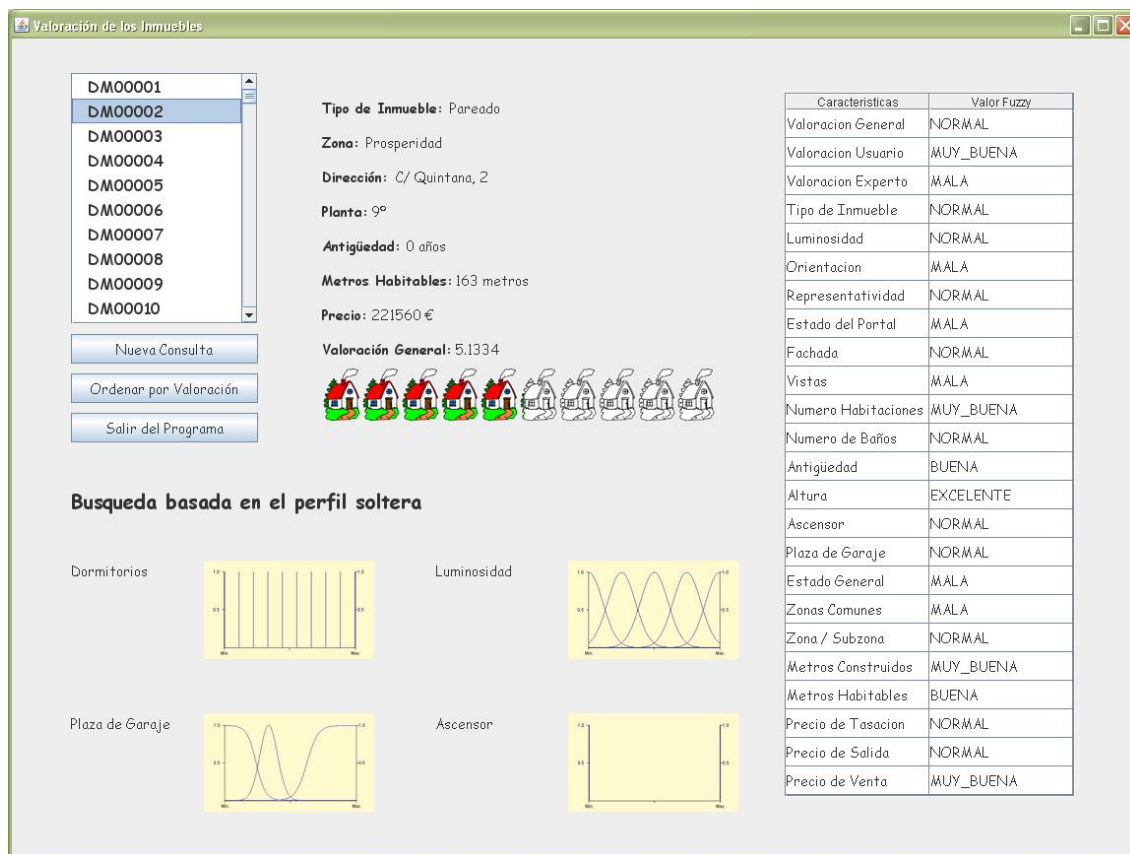


En ella, el comprador debe valorar la importancia del tipo de inmueble, de la orientación, de la representatividad y de las vistas.

La valoración de las características se hace de mayor a menor importancia mediante el siguiente rango de valores: Muy Importante, Bastante Importante, Importante, Poco Importante y Nada Importante.

Una vez que hemos evaluado la importancia que le da el comprador a esas características y que hemos elegido el límite del precio de salida, el motor de inferencia se encarga de calcular los inmuebles más apropiados para ese tipo de perfil.

Una vez evaluados los inmuebles mostramos todos aquellos que no sobrepasen el límite de precio fijado por el comprador. Los inmuebles aparecerán ordenados por su identificador, pero tenemos la posibilidad de poder ordenarlos según la valoración general obtenida.



En esta ventana se muestra las características más relevantes de los inmuebles obtenidos de la valoración para este perfil. Pinchando con el ratón sobre los identificadores de la izquierda vamos mostrando la información de cada inmueble. Sólo mostramos los inmuebles que no superan el límite del precio fijado por el usuario en la anterior ventana. En la parte inferior tenemos los conjuntos difusos que representan las distintas clases de pertenencia de las características primarias de ese perfil y en la parte de la derecha tenemos la valoración Fuzzy de todas las características del inmueble para este perfil.

Además de mostrarse por pantalla la valoración de los inmuebles para ese perfil, internamente, la aplicación genera dos ficheros en formato Excel, llamados **fichero_fuzzy.csv** y **fichero_nofuzzy.csv**, con las valoraciones generales de todos los inmuebles para ese determinado perfil.

Dentro de **fichero_nofuzzy.csv** encontramos un valor numérico que es la valoración general del inmueble, mientras que en **fichero_fuzzy.csv** no tenemos el valor

numérico, sino un valor fuzzy del inmueble que es un valor comprendido entre un rango de posibles valores. Los posibles valores que puede tener dicho rango son: Muy Mala, Mala, Normal, Buena, Muy Buena y Excelente.

Nota: En las versiones españolas de Microsoft Excel, los valores de las celdas del fichero `fichero_nofuzzy.csv` se muestran como un valor entero en vez de decimal, esto es debido a que en los archivos de formato csv no se puede especificar el tipo del campo. Para arreglarlo, tan solo habría que seleccionar todas las celdas y pulsar en la barra de menú sobre Formato → Celdas..., a continuación seleccionamos la categoría General y damos a Aceptar. De ésta manera cambiaremos el formato en que se muestran las celdas a decimal. Con la versión inglesa del producto no tendremos ese problema.

4.3.2. Resultados

Hemos realizado cuatro pruebas para ver los resultados que obtenemos. Para ello hemos cogido cada uno de los posibles perfiles que tenemos y hemos valorado los inmuebles para cada uno.

Cuando la aplicación pedía que valoráramos alguna característica del inmueble, hemos decidido siempre dar el valor Muy Importante a todas. Además hemos puesto el límite de precio en su valor máximo para así obtener todos los inmuebles que contiene la base de datos.

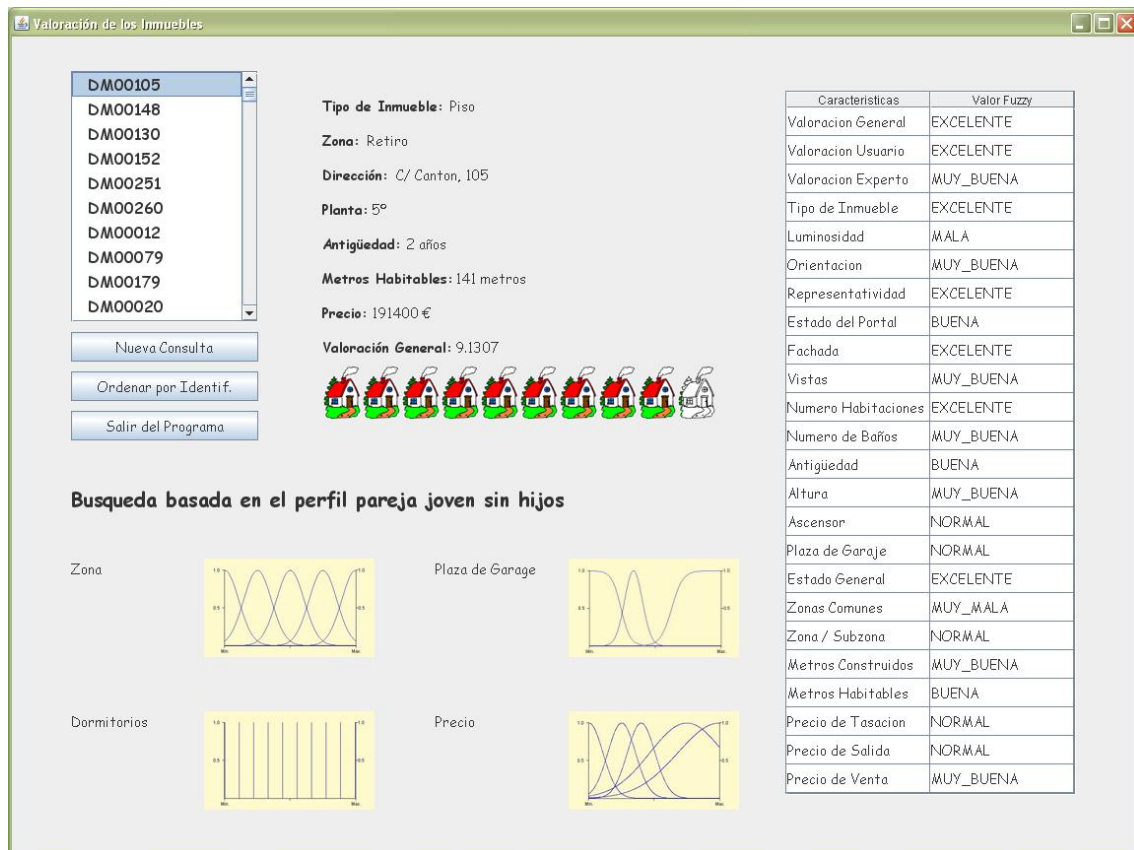
Para poder observar mejor los resultados obtenidos de realizar las distintas pruebas, hemos decidido ordenar los resultados de mayor a menor por su valoración general, que es la media aritmética de la valoración que hace el usuario más la valoración que hace el experto.

$$\text{Valoración General} = \frac{\text{Valoración Usuario} + \text{Valoración Experto}}{2}$$

Pareja joven sin hijos

En general, los resultados obtenidos son inmuebles con una valoración fuzzy de excelente o muy buena en cuanto a las características de tipo de inmueble y de representatividad, que son dos características valoradas por el perfil como muy importante. Diez de los quince primeros inmuebles tiene una valoración fuzzy de excelente en cuanto a la representatividad, mientras que los diez primeros inmuebles cuentan con una valoración de excelente respecto al tipo de inmueble.

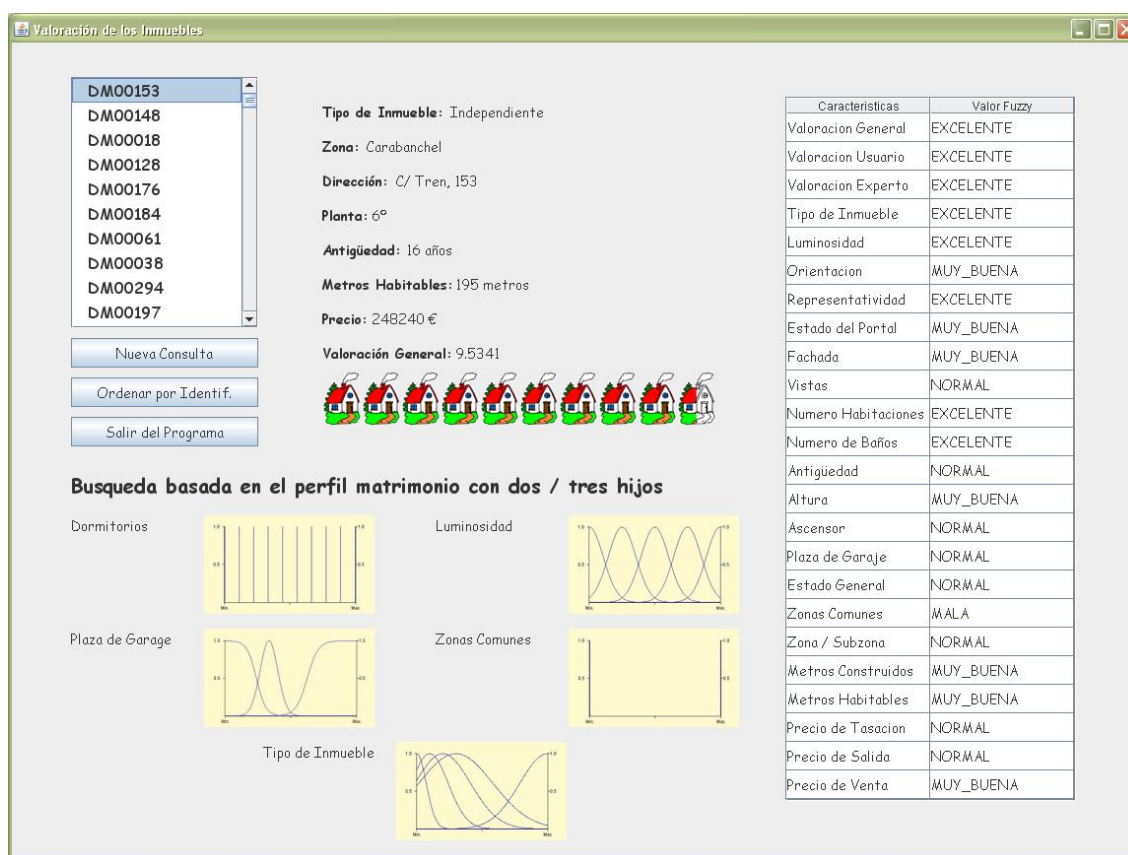
De los veinte primeros inmuebles, el 90 % cuenta con al menos una plaza de garaje y con dos o tres dormitorios, características primarias para perfil. Como podemos observar en la figura de debajo, el inmueble mejor valorado tiene un valor de 9.13.



Matrimonio con dos / tres hijos

Al observar los inmuebles vemos que, en general, el estado del portal y la luminosidad tienen una valoración fuzzy alta, ya que de los veinte primeros valores el 80 % tiene una valoración excelente respecto a la luminosidad.

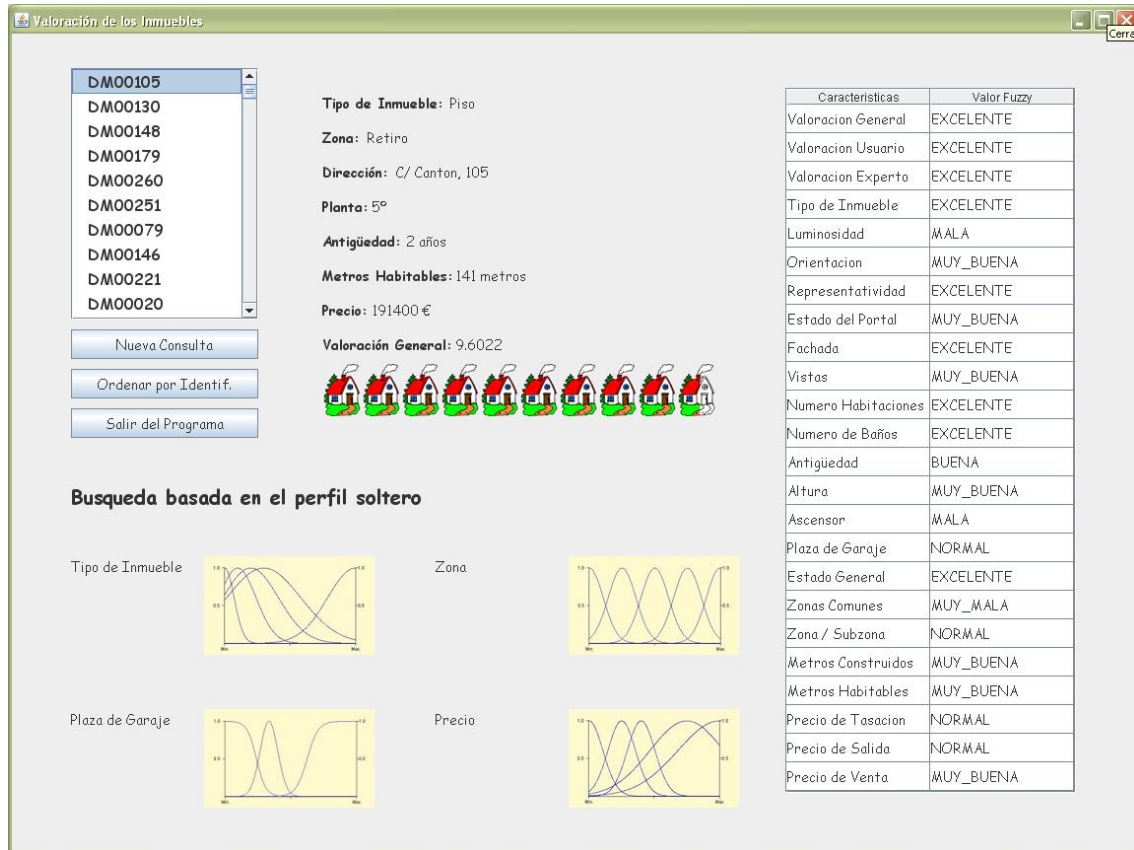
Además son inmuebles con dos o tres baños y con un alto número de dormitorios, cuatro o cinco dormitorios por inmueble, que es debido a que es una de las características primarias para éste perfil.



Hombre Soltero

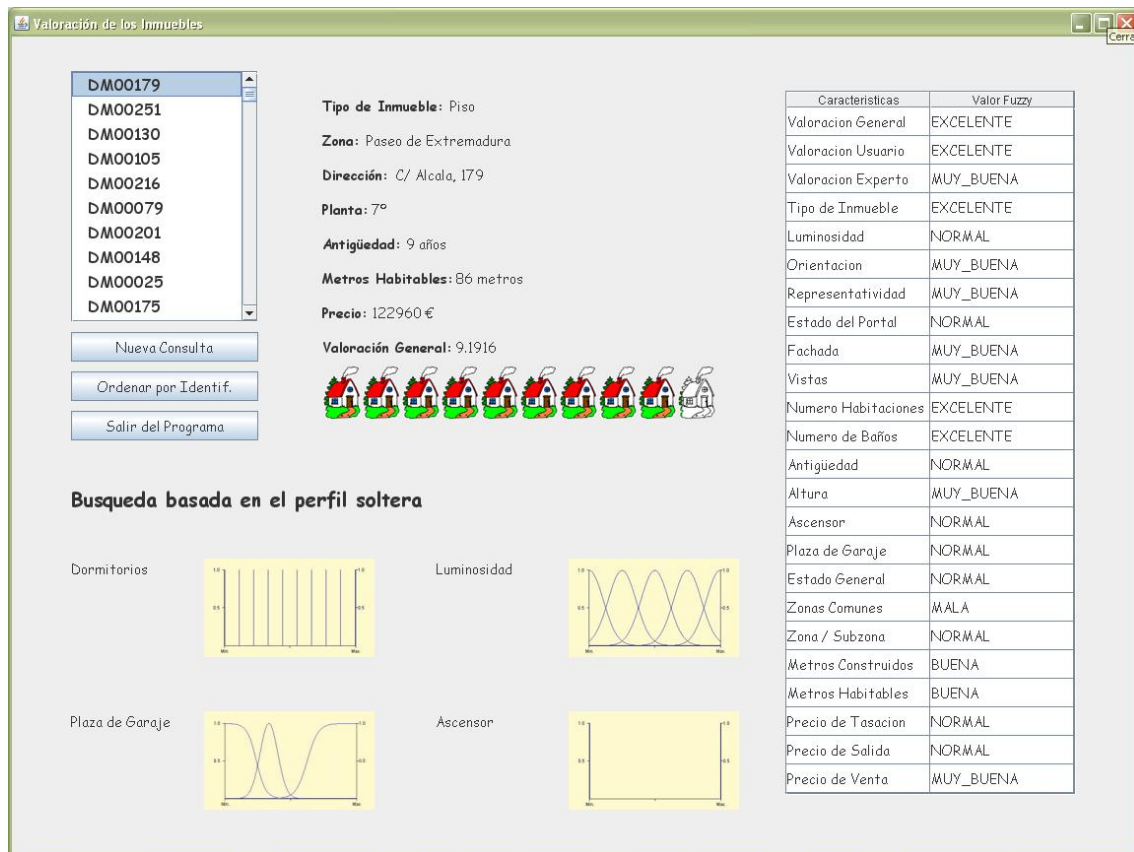
En el caso del soltero, los resultados obtenidos son muy parecidos a los que se obtienen de valorar la pareja joven sin hijos, esto es debido a que las características primarias para los dos perfiles son parecidas. Las diferencias obtenidas en los resultados son producidas gracias a la valoración que hace el usuario de las características mostradas en la segunda ventana.

Al haber elegido que la importancia de la representatividad, el estado del portal, la fachada y el estado general del inmueble sea muy importante, la mayoría de los inmuebles cuentan con un valor alto en dichas características, al menos tres de las cuatro características siempre tienen un valor muy buena o excelente.



Mujer Soltera

Por último tenemos la prueba de la mujer soltera. Al observar los principales inmuebles vemos que son todos pisos o adosados con una media de 150 metros cuadrados. El 90% de los inmuebles cuentan con al menos una plaza de garaje y dos o tres dormitorios, características primarias de todas aquellas personas solteras. Al haber puesto que la representatividad y las vistas son muy importantes, el 75% de los inmuebles cuentan con una valoración alta en dichos campos. Respecto al precio, observamos que el precio medio es de unos 200.000 euros, pero el inmueble mejor valorado tiene un precio inferior que está lejos de dicha cantidad.



4.4 Trabajo Futuro

Desde nuestro punto de vista podrían hacerse dos ampliaciones en nuestra aplicación.

Una primera ampliación en cuanto a funcionalidad que creemos resultaría interesante sería añadir la posibilidad de parametrizar un perfil genérico en el que el usuario pudiera elegir la importancia de todas las características del inmueble que deseara. También sería apropiado y no muy costoso introducir el concepto de valoración de inmuebles en alquiler. De esa forma podríamos cubrir diferentes perfiles que puedan buscar funcionalidades distintas.

Otra posible ampliación podría ser integrar la aplicación en una página web para poder acceder a ella a través de la red, de esta forma la aplicación sería más accesible, y las posibilidades que se abren son muy interesantes, en forma de interconexión con otros servicios o más concretamente en la adquisición de fuentes de conocimiento muy amplias y totalmente reales.

Desde un punto de vista más técnico, mejorar el sistema CAD X-Fuzzy, manteniendo el sistema gráfico de modelado y creando un API de lógica difusa bajo Java en vez de código generado una vez creado el modelo.

Por otro lado, el refinamiento del cálculo de la valoración de las características de los inmuebles siempre es algo presente, dada la evolución del campo de la lógica difusa, estudiando la aplicación de nuevas técnicas y métodos.

5. Conclusiones

El sistema de modelador CAD de lógica difusa X-Fuzzy nos ha resultado realmente bueno para especificar el sistema pero el código generado no está a la altura de las circunstancias por los siguientes motivos:

- No es un API, lo que imposibilita empezar a integrar el sistema hasta no tenerlo acabado;
- Falta de estructuración del código generado. Solo genera 4 clases genéricas, por lo que es muy poco reutilizable y poco flexible;
- Si hay que realizar algún cambio es necesario recompilar todo el código;
- No tiene ningún sistema de chequeo de precondiciones, por lo que si se introduce un valor fuera del rango del tipo de valor utilizado en el sistema, lanza una excepción genérica. Creemos que esto lo debería realizar el propio Xfuzzy;
- No lanza ningún tipo de excepciones propias, todas genéricas de java, por lo que es difícil depurarlo.

Lo ideal sería tener el sistema de modelado de Xfuzzy y por debajo, a nivel de código, un API de lógica difusa y que Xfuzzy generara el código en java para utilizar directamente el API. Respecto a otras cosas:

- Los valores del experto deberían ser más precisos (son 0, 1 o 2) y deberían estar en un rango de 0 a 10 para que los resultados fueran algo más ajustados;
- Ciertos elementos, como tipo de vivienda o precio deberían ser filtros excluyentes y no un elemento más a ponderar;
- Los cambios introducidos por el usuario en las importancias son poco influyentes en el resultado final puesto que hay muchos elementos a valorar, y el cambio es poco significativo en la valoración general.

6. Manual del Usuario

Contenido del CD

El contenido del CD que adjuntamos se encuentra dividido en cuatro carpetas: Archivos Adjuntos, Código, Documentación y Ejecutable.

En la carpeta Código podemos encontrar el código Java del proyecto. Para poder ejecutar este código es necesario el entorno de programación Eclipse.

En la carpeta Documentación se encuentra la memoria del proyecto, así como el motor de inferencia generado en Xfuzzy.

En la carpeta Ejecutable tenemos un archivo .jar para ejecutar la aplicación sin la necesidad de Eclipse.

Por último tenemos la carpeta Archivos Adjuntos, en ella se encuentran varias aplicaciones. Un JDK 1.6 necesario para la ejecución de la aplicación, así como el entorno de programación Eclipse.

Además de estos archivos, en la carpeta encontramos la aplicación Xfuzzy 3.0 para poder ver el motor de inferencia.

Instalación del Software

Antes de ejecutar nuestra aplicación, el usuario debe comprobar que tiene instalada la Máquina Virtual de Java en su ordenador, con un JDK con versión 1.5 o superior.

Si el usuario no la tuviera instalada, podría encontrarla en la carpeta Archivos Adjuntos o si lo desea podría descargarla de <http://java.sun.com>.

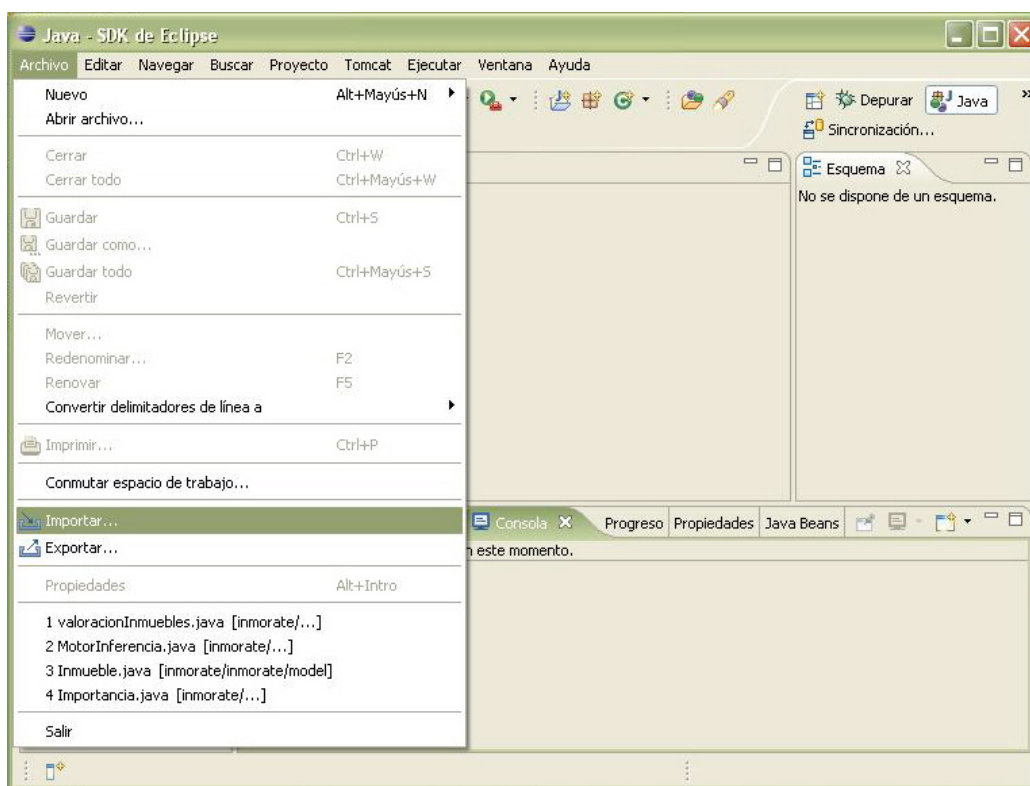
En el caso que el usuario deseara ejecutar la aplicación desde Eclipse, deberá comprobar que lo tiene instalado. En caso que no lo tenga podrá encontrarlo también en

la carpeta Archivos Adjuntos o bien en la página de descargas de Eclipse:
<http://www.eclipse.org>.

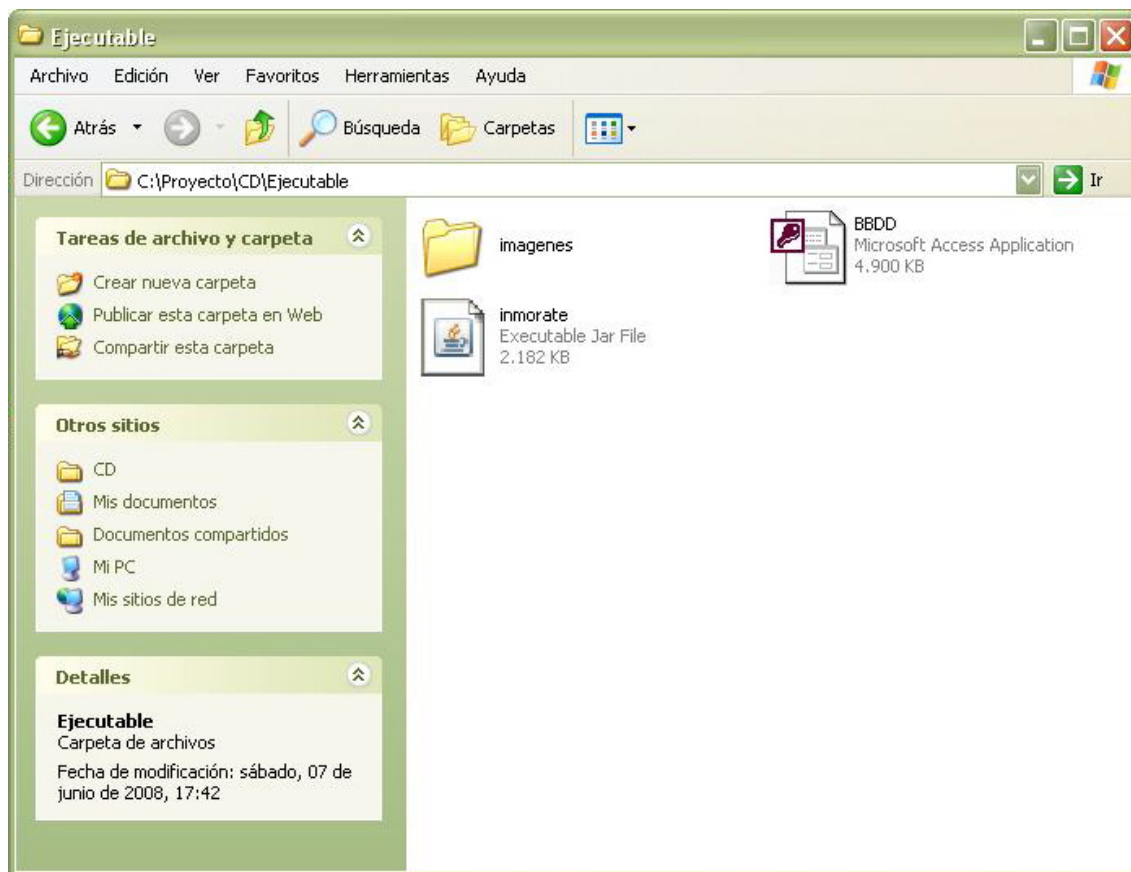
Manual de Usuario

El usuario tiene dos formas de ejecutar la aplicación:

- Puede abrir el entorno de programación Eclipse, importar el proyecto que se encuentra en la carpeta Código y ejecutar la clase **Start.java** que se encuentra en el paquete **inmorate.controlador**.



- O bien puede ejecutar directamente la aplicación que se encuentra en la carpeta Ejecutable.



Nota: En caso de querer utilizar el ejecutable. Si se desea copiar a otro directorio, se deben copiar todos los archivos que se encuentran en la carpeta Ejecutable.

La aplicación está optimizada para una resolución de 1152 x 864 píxeles por pulgada, por lo que es recomendable ajustar la resolución de su monitor a una resolución similar o superior.

Una vez que tenemos ejecutando la aplicación, aparece en pantalla una ventana de bienvenida con seis botones. Los dos de la parte superior se utilizan para elegir el idioma deseado (español o inglés) y los cuatro de la parte inferior para elegir el tipo de perfil del usuario.

Una vez que hemos elegido el idioma, automáticamente al elegir el perfil al cual el comprador pertenece, vamos a otra ventana intermedia que dependerá del perfil

seleccionado. En estas ventanas aparecen una serie de características del inmueble, que son diferentes para cada ventana, las cuales el usuario tendrá que valorarlas de Muy Importante a Nada Importante según el grado de importancia que tengan dichas características para él.

Valore de mayor a menor importancia:

Tipo de Inmueble

Bastante Importante	▼
Muy Importante	
Bastante Importante	
Importante	
Poco Importante	
Nada Importante	

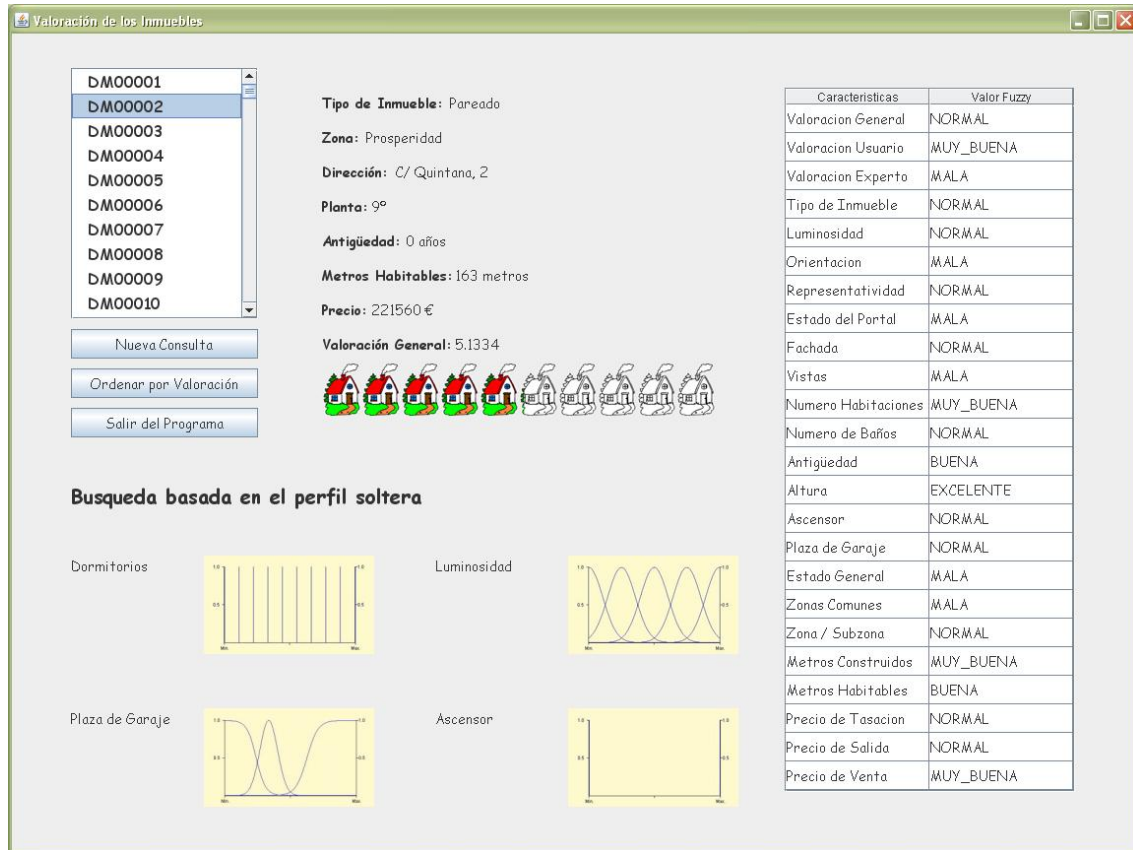
En esta ventana disponemos además de una barra deslizadora para poner el límite del precio de salida del inmueble. Podemos desplazar la barra entre un valor mínimo de 100.000 € y uno máximo de 500.000 €.

200.000 €

100.000 € 500.000 €

Cuando hemos terminado de valorar las características debemos pulsar sobre el botón **Valorar** para que la aplicación valore los inmuebles según los parámetros establecidos por el usuario. En el caso que deseemos volver a la pantalla inicial, el usuario deberá pulsar sobre el botón **Cancelar**. Una vez que se que hayamos valorado los distintos inmuebles, se mostrará en la parte inferior de la ventana el resultado de la valoración general, de la valoración del usuario y de la valoración del experto para cada inmueble y además se activará el botón de **Continuar...**

Si pulsamos sobre el botón Continuar vamos a una ventana donde tendremos los resultados obtenidos para ese tipo de perfil especificado en la pantalla de bienvenida.



Como podemos observar en la imagen, en la parte de la izquierda aparecen los identificadores de los inmuebles. Pulsando sobre cualquier identificador obtenemos las características más relevantes del inmueble: El tipo de inmueble, la dirección, el precio...

Dado que el comprador ha elegido un límite del precio del inmueble en la ventana de bienvenida, sólo se muestran los inmuebles que no superen ese límite impuesto.

Por último, en la parte inferior izquierda se encuentran tres botones: **Nueva Consulta**, **Ordenar por Valoración** y **Salir del Programa**. El botón de "Nueva Consulta" muestra la ventana inicial para que el usuario pueda realizar una nueva consulta y el botón "Salir del Programa" termina la ejecución del programa. En un

principio los resultados aparecen ordenados por el identificador pero en el caso que el usuario lo desee, pulsando sobre el botón **Ordenar por Valoración**, la aplicación mostrará los resultados obtenidos ordenados por la valoración general del inmueble.

7. Apéndices

7.1. Instalación de XFuzzy 3.0

Requisitos del Sistema

Xfuzzy 3.0 puede ser ejecutado sobre cualquier plataforma que disponga del "Java Runtime Environment" (JRE). Para definir nuevos paquetes de funciones es también necesario disponer de un compilador Java. La última versión del "Java Software Development Kit", incluyendo el JRE, un compilador Java y otras herramientas relacionadas, puede encontrarse en <http://java.sun.com/j2se/>.

Guía de Instalación:

1. Descargar el fichero XfuzzyInstall.jar de la dirección:
http://www.imse.cnm.es/Xfuzzy/Xfuzzy_3.0/download_sp.html
2. Ejecutar este fichero. Sobre MS-Windows basta con pulsar sobre el icono. En general el fichero se ejecuta con el comando "java -jar XfuzzyInstall.jar". Esto abrirá la siguiente ventana de instalación.



3. Elegir un directorio para instalar Xfuzzy. Si el directorio no existe se creará en el proceso de instalación.

4. Elegir el directorio de los ejecutables de java (java, javac, jar, etc.). Este directorio suele ser el subdirectorio "/bin" de la instalación del J2SE.
5. Elegir un navegador para mostrar los ficheros de ayuda.
6. Pulsar en el botón "Install" para descomprimir la distribución de Xfuzzy en el directorio base seleccionado.
7. Los ejecutables de Xfuzzy residen en el directorio "/bin".
8. Los ficheros ejecutables son ficheros de comandos. Si se cambia la localización de la distribución de Xfuzzy deberá repetir el proceso de instalación.

7.2. Instalación de Eclipse

Requisitos del Sistema

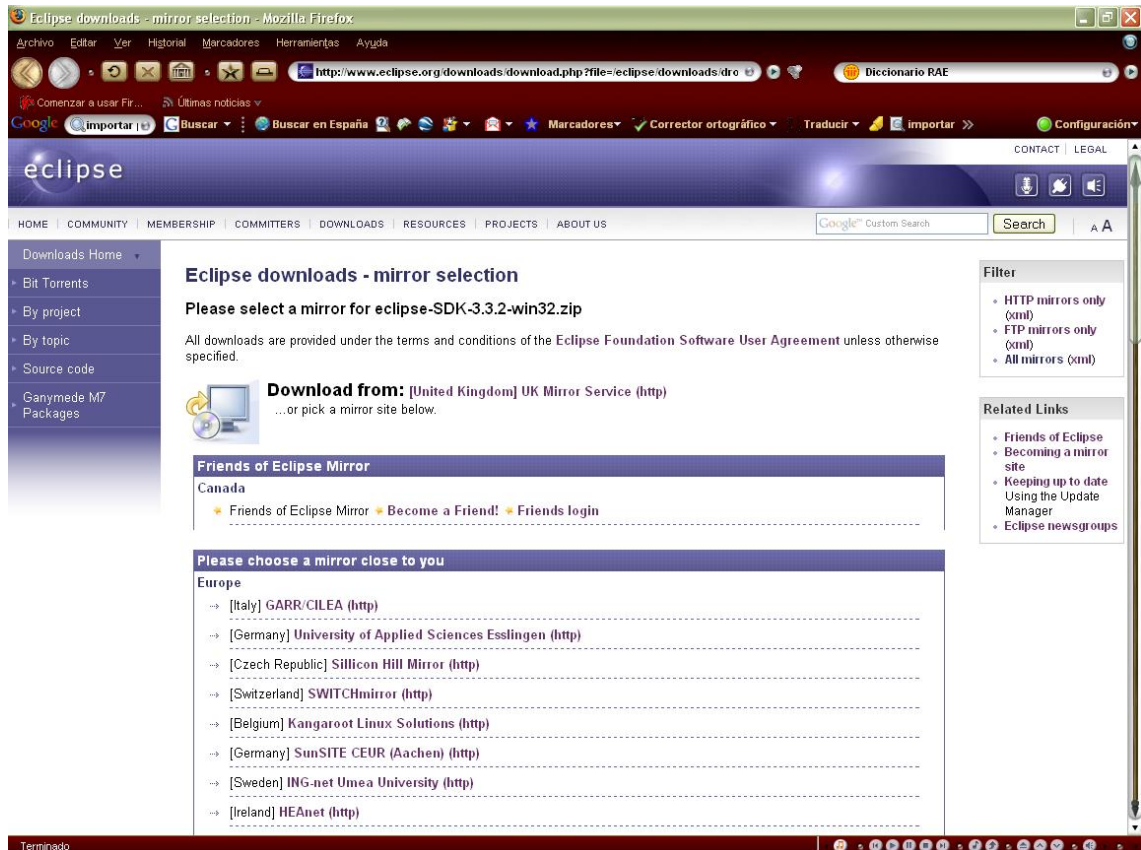
Los requerimientos necesarios es tener instalado el JDK 1.3 o superior, tener alrededor de 300 MB de espacio libre en el disco duro y preferiblemente tener 256 MB de memoria.

Guía de Instalación:

Lo primero que debemos hacer es descargarse el software correspondiente desde la página de eclipse: <http://www.eclipse.org>. Para ello vamos a la página de descargas de eclipse y nos descargamos la última versión del programa que se encuentra en la dirección:

<http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.3.2-200802211800/eclipse-SDK-3.3.2-win32.zip>

En dicha página seleccionamos un "Mirror" y a continuación comenzará la descarga del programa.



Eclipse puede funcionar en Windows, Linux, Solaris, etc...

Descomprimos el fichero eclipse-SDK-3.3.2-win32.zip que nos acabamos de descargar en el directorio c:\eclipse.

Por último para arrancar la plataforma eclipse basta con ejecutar eclipse.exe que se encuentra dentro de dicho directorio.

8. Palabras Clave

Lista de palabras clave:

- Valoración
- Inmueble
- Clasificación por perfiles
- Buscador de producto
- Sistema de recomendación
- Lógica fuzzy

9. Bibliografía

\bibitem{XFUZZY} XFuzzy 3.0. Centro Nacional de Microelectrónica.

Instituto de Microelectrónica de Sevilla.

<http://www.imse.cnm.es/xfuzzy>

[FLINS06]

Victoria López, J. Miguel Cleva y Javier Montero, "A functional tool for Fuzzy First Order Logic Evaluation", D. Ruan, P. D'hont, P.F.

Fantoni, M. De Cock, M. Nachtegael and E.E. Kerre, eds. Applied Artificial Intelligence, World Scientific, New Jersey, 2006; pp.19--26.

[CEDI07]

Victoria López y Javier Monero, "Análisis de requisitos difusos en valoración de inmuebles", II Simposio sobre Lógica Fuzzy y Soft Computing, II Congreso Español de Informática, Zaragoza, Septiembre, 2007; pp: 181-187

[FLINS08]

Victoria López, Álvaro del Monte y Javier Montero, "Fuzzy logic in real estate valuation", Computational Intelligent in Decisión and Control, World Scientific, 2008; pp.1021-1026.

Curso Introductorio de Conjuntos y Sistemas Difusos por el Dr. José Galindo G. de la Universidad de Málaga (España):

<http://www.lcc.uma.es/~ppgg/FSS/>

Página de Xfuzzy 3.0:

http://www.imse.cnm.es/Xfuzzy/Xfuzzy_3.0/index.html